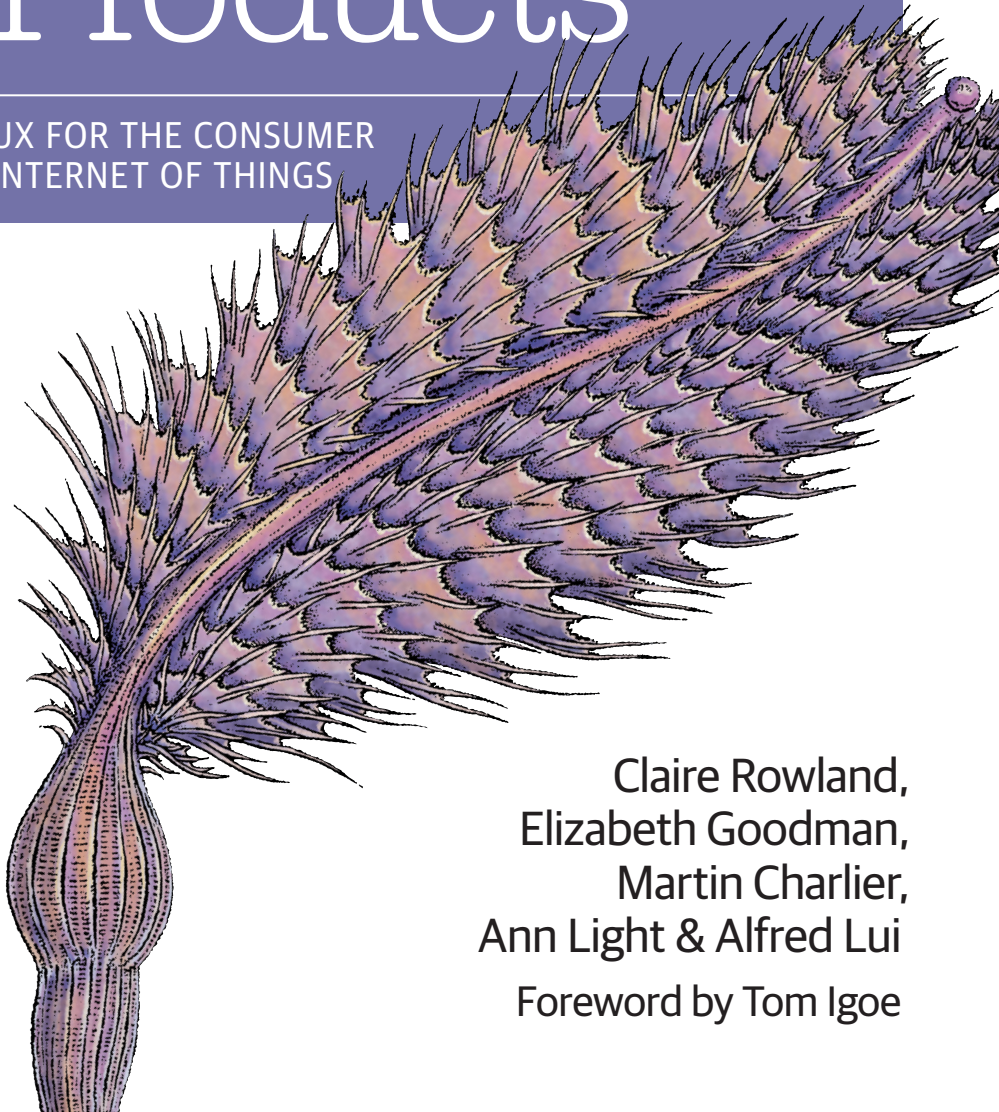


O'REILLY®

# Designing Connected Products

UX FOR THE CONSUMER  
INTERNET OF THINGS



Claire Rowland,  
Elizabeth Goodman,  
Martin Charlier,  
Ann Light & Alfred Lui  
Foreword by Tom Igoe

# Designing Connected Products

Networked thermostats, fitness monitors, and door locks show that the Internet of Things can (and *will*) enable new ways for people to interact with the world around them. But designing connected products for consumers brings new challenges beyond conventional software UI and interaction design.

This book provides experienced UX designers and technologists with a clear and practical roadmap for approaching consumer product strategy and design in this novel market. By drawing on the best of current design practice and academic research, *Designing Connected Products* delivers sound advice for working with cross-device interactions and the complex ecosystems inherent in IoT technology.

## Topics Include:

- How the technology of IoT affects UX
- Product and design strategy for connected devices
- Industrial design
- Interface and interaction design for embedded devices
- Cross-device interactions and interusability
- Interoperability
- Responsible IoT design
- Designing with data
- Prototyping and user research methods for connected products

“This is more than a UX book; it covers all of the critical design and technology issues around making great connected products.”

—David Rose

Entrepreneur, MIT Media Lab researcher  
and author of *Enchanted Objects*

“Whether you’re an IoT pro or just getting started designing connected products...this book takes a clear-eyed look at IoT from all angles.”

—Dan Saffer

author of *Microinteractions*

“This book pretends to be a primer on designing the Internet of Things (and it’s an excellent one), but quickly reveals itself as a primer on nearly every aspect of contemporary design.”

—Matt Jones

Interaction Design Director, Google  
Creative Lab and former principal at BERG

UX/DESIGN

US \$49.99

CAN \$57.99

ISBN: 978-1-449-37256-9



9 781449 372569



Twitter: @oreillymedia  
facebook.com/oreilly

# Praise for *Designing Connected Products*

*“This is more than a UX book; it covers all of the critical design and technology issues around making great connected products.”*

**DAVID ROSE—ENTREPRENEUR, MIT MEDIA LAB RESEARCHER,  
AND AUTHOR OF *ENCHANTED OBJECTS***

*“Whether you’re an IoT pro or just getting started designing connected products, this comprehensive book has something for everyone, from examinations of different network protocols all the way up to value propositions and considerations for hardware, software, and services. This book takes a clear-eyed look at IoT from all angles.”*

**DAN SAFFER—AUTHOR OF *MICROINTERACTIONS***

*“This book should be on the desk of anyone contemplating an IoT product. Covering the end-to-end ecosystem of devices through to backend services, the authors address so many aspects of the UX journey that constantly make you think: ‘Oh, yes, really good point!’ There is comprehensive coverage of the technology options available as building blocks and patterns for an IoT solution. The book skillfully delivers a wealth of sound UX wisdom, backed up by of-the-moment examples, which will resonate with practitioners already familiar with the IoT world, and will serve as a modern educational text for everyone.”*

**DR. ANDY STANFORD-CLARK—IBM DISTINGUISHED  
ENGINEER FOR INTERNET OF THINGS**

*“The authors do a fantastic job of not only exposing the complexities of a future pervaded by connected things, but suggesting exactly how we can consciously go about preventing that complexity from overwhelming us. This book will be an essential reference for designers as they come to grips with creating these new type of things that will live throughout our lives.”*

**BEN FULLERTON—CREATIVE DIRECTOR, SONOS**

*“The explosion of possibilities with the Internet of Things will only come about if people can understand and use it. Designing Connected Products brings much needed clarity to the challenge and provides a toolkit to build better solutions. It also acts as a showcase of the brightest and best Internet of Things projects and products.”*

**ADRIAN MCEWEN—FOUNDER, MCQN LTD., AND  
AUTHOR OF DESIGNING THE INTERNET OF THINGS**

*“In a field where the hype can change even faster than the technology, this book grounds designers, entrepreneurs, and technologists in what matters—the fundamentals of people’s behavior, networked technologies, and the context both now find themselves in. In doing this it reaches back to a larger historical field of research, design, and evaluation that is invaluable.*

*As a grizzled veteran of several campaigns within the matter-battle of the Internet of Things, I was pleasantly surprised to find the number of times this book made me pause, think, and rethink my own work (and that of others). A very valuable addition to the canon of design thinking in this emerging area.*

*This book pretends to be a primer on designing the Internet of Things (and it’s an excellent one) but it reveals itself quickly as really being a primer on nearly every aspect of contemporary design—as the Internet touches nearly every aspect of it.”*

**MATT JONES—INTERACTION DESIGN DIRECTOR,  
GOOGLE CREATIVE LAB, AND FORMER PRINCIPAL AT BERG**

*“Usability is THE challenge for the Internet of Things. This book is full of excellent guidance on how to achieve that most elusive but essential quality of an IoT product: simplicity. This book shows you how to delight your connected product customers by treating their attention as the precious and finite resource that it is.”*

**PILGRIM BEART—FOUNDER, ALERTME**

*“In an industry that is dominated by engineering concerns, this book will illuminate the human side of IoT. It’s a broad and deep survey with a rich set of examples for every point.”*

**MATT BIDDULPH—COFOUNDER, THINGTON**

# Designing Connected Products

*UX for the Consumer Internet of Things*

Claire Rowland, Elizabeth Goodman, Martin Charlier,  
Ann Light, and Alfred Lui

**O'REILLY®**

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

## Designing Connected Products

by Claire Rowland, Elizabeth Goodman, Martin Charlier, Ann Light, and Alfred Lui

Copyright © 2015 Rowland, Goodman, Charlier, Light, Lui. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc.,  
1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<https://www.safaribooksonline.com/>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

<b>Editors:</b> Mary Treseler and Angela Rufino	<b>Cover Designer:</b> Ellie Volckhausen
<b>Technical Editor:</b> Ann Light	<b>Interior Designers:</b> Ron Bilodeau and Monica Kamsvaag
<b>Production Editor:</b> Kara Ebrahim	<b>Illustrator:</b> Rebecca Demarest
<b>Copyeditor:</b> Jasmine Kwityn	<b>Compositor:</b> Kara Ebrahim
<b>Proofreader:</b> Kim Cofer	
<b>Indexer:</b> Ron Strauss	

May 2015: First Edition.

### Revision History for the First Edition:

2015-05-08	First release
2015-07-10	Second release

See <http://www.oreilly.com/catalog/errata.csp?isbn=0636920031109>  
for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Designing Connected Products*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

ISBN: 978-1-4493-7256-9

[LSI]

*For Silas, who will never know an Internet  
without all the Things*





# [ Contents ]

<i>Foreword</i> .....	xiii
<i>Preface</i> .....	xvii
<b>Chapter 1</b> What's Different About User Experience Design for the Internet of Things? .....	1
<i>by Claire Rowland</i>	
How Is UX for IoT Different? .....	4
A Design Model for IoT .....	17
Summary .....	27
<b>Chapter 2</b> Things: The Technology of Connected Devices .....	29
<i>by Claire Rowland</i>	
Types of Connected Device .....	30
Multipurpose Computers .....	31
Bridging Physical and Digital: Sensors and Actuators .....	48
The Challenge of Powering Devices .....	53
Conserving Battery Life .....	53
Summary .....	56
<b>Chapter 3</b> Networks: The Technology of Connectivity .....	57
<i>by Claire Rowland</i>	
Why Is Networking Relevant to IoT UX? .....	58
Networking Issues That Cause UX Challenges for IoT .....	59
The Architecture of the Internet of Things .....	66
Types of Network .....	75
Network Communication Patterns .....	87

Internet Service.....	92
Summary.....	100
<b>Case Study 1 Proteus Digital Health: The Connected Pill .....</b>	<b>103</b>
<i>by Arna Ionescu</i>	
<b>Chapter 4 Product/Service Definition and Strategy.....</b>	<b>111</b>
<i>by Claire Rowland</i>	
Making Good Products.....	112
From Innovation to Mass Market.....	116
Tools Versus Products .....	123
What Makes a Good Product? .....	128
Services in IoT.....	135
Business Models.....	144
Summary .....	149
<b>Chapter 5 Understanding People and Context.....</b>	<b>151</b>
<i>by Elizabeth Goodman</i>	
The Role of Research in Connected Product Design .....	152
Initial Questions and Concepts .....	153
Techniques: from Asking to Watching to Making ..	169
Summary.....	181
<b>Chapter 6 Translating Research into Product Definitions .....</b>	<b>183</b>
<i>by Elizabeth Goodman</i>	
Generating the Elevator Pitch .....	184
Why Does Your Product Matter? .....	185
What Is Your Product?.....	195
What Does the Product Do? .....	199
Recurring Questions for Product Strategy .....	207
Summary.....	212
<b>Case Study 2 Little Kelham: Connected Home .....</b>	<b>215</b>
<i>by Matt Edgar</i>	

<b>Chapter 7</b>	<b>Embedded Device Design</b> .....	<b>221</b>
<i>by Martin Charlier</i>		
	An Introduction to Thinking About Physical Objects in IoT.....	222
	Making Stuff: Differences to UX.....	236
	Essentials of the Design Process .....	241
	Three Faces of a Physical Product .....	256
	Summary.....	272
<b>Chapter 8</b>	<b>Interface and Interaction Design</b> .....	<b>275</b>
<i>by Martin Charlier</i>		
	Types of Interaction.....	276
	IoT-Specific Challenges and Opportunities.....	312
	Universal Design and Accessibility.....	320
	Summary.....	324
<b>Case Study 3</b>	<b>Ford SYNC 3: Connected Car</b> .....	<b>327</b>
<i>by Parrish Hanna</i>		
<b>Chapter 9</b>	<b>Cross-Device Interactions and Interusability</b> .....	<b>337</b>
<i>by Claire Rowland</i>		
	Cross-Platform UX and Usability.....	338
	What Is Interusability? .....	340
	Conceptual Models and Composition .....	341
	Consistency.....	360
	Continuity.....	368
	Summary.....	380
<b>Chapter 10</b>	<b>Interoperability</b> .....	<b>381</b>
<i>by Claire Rowland</i>		
	The CompuServe of Things .....	382
	What Is Interoperability and Why Is It a Problem? ..	385
	How Can Devices Interoperate? .....	393
	How Can We Improve Interoperability?.....	401
	The UX of Interoperability .....	405
	Summary.....	411

<b>Case Study 4 LOOP: Connected Pelvic Floor Exerciser</b> .....	<b>413</b>
<i>by Adam Scheuring</i>	
<b>Chapter 11 Responsible IoT Design</b> .....	<b>419</b>
<i>by Ann Light and Claire Rowland</i>	
Security.....	420
Privacy.....	436
Social Engineering .....	450
Environment.....	457
Summary.....	464
<b>Chapter 12 Supporting Key Interactions</b> .....	<b>467</b>
<i>by Claire Rowland and Martin Charlier</i>	
Setup.....	469
In-Life Housekeeping .....	479
Discovery .....	486
Control Experiences .....	487
Platforms .....	494
The Technology of Getting Things Connected .....	499
Summary.....	517
<b>Case Study 5 BRCK: Rugged Portable WiFi Hotspot</b> .....	<b>519</b>
<i>by Jeff Maina</i>	
<b>Chapter 13 Designing with Data</b> .....	<b>527</b>
<i>by Alfred Lui and Claire Rowland</i>	
Introduction .....	527
Data in IoT.....	528
Types of Data-Driven Product .....	544
What This Means for Design .....	550
Summary.....	573

<b>Chapter 14</b>	<b>Iterative Design: Prototyping and Learning</b> .....	575
<i>by Elizabeth Goodman</i>		
	The Necessity of Working Iteratively .....	576
	Using Prototypes to Answer Questions .....	577
	How Do You Decide What to Prototype? .....	603
	Evaluation: Success Demands Failure .....	607
	Summary .....	608
<b>Chapter 15</b>	<b>Designing Complex, Interconnected Products and Services</b> .....	611
<i>by Claire Rowland</i>		
	It's Complicated... ..	612
	Scaling the UX .....	617
	Control .....	624
	Approaches to Managing Complexity .....	653
	Summary .....	666
<b>Appendix A</b>	<b>Companies, Products, and Links</b> .....	667
	<b>Index</b> .....	677



## [ *Foreword* ]

Since the dot-com crash of the early 21st century, the Internet has grown up. No longer a just a collection of virtual storefronts, it now facilitates a wide range of human expression and interaction. It catalyzes changes in our culture. It supports intimate contact, and it fuels revolution. Through it, we've realized a shift in our relationships, from a centralized model where a few central players control how we communicate to a decentralized one in which the size of your organization matters less than the relevance of your message. The decentralization of media and communication is the most important change brought about by the Internet.

Thus far, we've effected that change only through glass rectangles, but not for long. The Internet now connects your car, your car keys, your door lock, your lights, and your exercise regimen. Your informational body can now have a more direct effect on your physical body, and vice versa. We're only just beginning to understand the implications of this, and to develop the tools to design for it.

Like the authors, I'm not a fan of the term "Internet of Things." It overlooks this major change that the Internet has brought about in how we communicate, and instead, presents an ideal world in which the automation and centralization of data collection comes first, and the interaction of people and communities comes a distant second, if at all. This change is not about the things, but about the physical interaction those things enable.

We remember (and use) great products because of the experiences they help us to realize. Our interaction with a product—and with each other through it—is what makes it memorable. Really great connected products will be memorable for the same reason. We won't remember them

because of the data collected, but because of how they will enhance our lives and our connections to each other. Designing connected products is not just an engineering problem. This job takes creative input from many different disciplines: engineering, industrial design, anthropology, and user experience design, to name a few.

In this book, Claire Rowland and her colleagues outline a thorough framework for practicing the design of connected products and services. They provide the overview of the technical landscape that you'd expect from such a book, and best practices for designing connected products, including user research, business model formation, and prototyping methods, of course. But they also tackle issues that aren't often covered: two chapters on understanding your fellow team members, for example. By juxtaposing the assumptions of each discipline, from industrial design to UX design to anthropology to product engineering, they provide insight on why something that seems nonsensical to an engineer might be important to a designer, and how the concerns of all team members are necessary to do a thorough job.

Looking beyond the everyday production tasks, they introduce two critical concepts that anyone working in this area needs to understand: interoperability and interusability. Without interoperability, we are doomed to a future of incompatible devices and standards from competing brands. Without interusability, we are doomed to a future of relearning core user experience concepts like connect, play, stop, and restart over and over again with each new interface.

A world in which connected devices are ubiquitous is one in which privacy, control, and transparency are radically changed. As designers of these systems, we need to take responsibility for our role in this shift. The authors acknowledge this, and provide an overview of the ethical considerations of designing connected products. This is one of the most valuable parts of the book, as it helps designers to think not just about the product or the business model, but the impact of their work on customers' safety, privacy, and well-being. Gathering all the data in the world is pointless if we sacrifice quality of life for it.

I've been building network-connected physical interfaces, and teaching students how to do it, since 2001, and this is the first book I've encountered that gives designers a detailed and practical approach to do this.



There are plenty of technical manuals (I've written a few) and there are some excellent critical monographs, but there hasn't been a great design manual for connected devices until now. Thank you, Claire, Ann, Martin, Liz, and Alfred for making my job easier through this book.

**TOM IGOE**



# [ Preface ]

**BY CLAIRE ROWLAND**

*My grandfather could probably have told you how many electric motors he owned. There was one in the car, one in the fridge, one in his drill and so on.*

*My father, when I was a child, might have struggled to list all the motors he owned (how many, exactly, are in a car?) but could have told you how many devices were in the house that had a chip in.*

*Today, I have no idea how many devices I own with a chip, but I could tell you how many have a network connection. And I doubt my children will know that, in their turn.*

**BENEDICT EVANS<sup>1</sup>**

## From “Internet of Things” to “Connected Products”

Most of you who are old enough to be reading this book will have first experienced the Internet on a personal computer (PC). Going online was a special activity, done sitting down at a desk. In time, computers became smaller and more portable. Since early 2014, mobile Internet usage has outstripped that on PCs.<sup>2</sup> Most of us carry at least one Internet device with us all the time. The services and content it provides us are an intrinsic part of the fabric of daily life. But still, mostly, it is something we look at through glowing rectangular screens. That is now changing.

---

<sup>1</sup> <http://ben-evans.com/benedictevans/2014/5/26/the-internet-of-things>

<sup>2</sup> <http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report>

As I write, analysts are engaged in a PR race to forecast ever larger numbers of devices on the Internet, from Gartner's 26 billion devices by 2020 to Ericsson's 50 billion by the same year.<sup>3,4</sup>

Technology pioneer Kevin Ashton coined the term "Internet of Things" in 1999, while proposing supply chain management improvements to his then employer, Procter & Gamble. Using radio-frequency identification (RFID) to identify and track products automatically would save a huge amount of human work entering data into computers. Ashton saw this as part of a paradigm shift from computers learning about the world entirely from data originated by people, to being able to gather their own data:

If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.<sup>5</sup>

These days, the term "Internet of Things" (or IoT) is commonly used to encompass a much broader spectrum of technology. IoT now does not just mean things that can be identified, but things with onboard computation, network connections, and the power to sense the environment and act on the physical world—sometimes even autonomously.

As a label, IoT isn't perfect. It says nothing of the people who are also a fundamental part of the network.<sup>6</sup> It suggests that this is a new type of Internet, rather than an important extension of the Internet we already have. Some in the field prefer alternative terms with different shades of meaning, such as "ubiquitous computing/ubicomp," "pervasive computing," "connected devices," "smart objects," "programmable world," "Web of Things," or the "Internet of Everything."

---

3 <http://www.gartner.com/newsroom/id/2636073>

4 <http://www.ericsson.com/thecompany/press/releases/2010/04/1403231>

5 Kevin Ashton, "That 'Internet of Things' Thing," *RFID Journal*, July 22, 2009, <http://www.rfidjournal.com/articles/view?4986>.

6 Although Ashton himself frames IoT in terms of its human benefits, see previous reference.

In this book, we'll skip the debate and use the term "IoT" for pragmatic reasons. It has the most traction in the commercial world. In time, the term will probably fade from favor, but the technology will continue to develop. One day, we will consider connected devices to be part of the mainstream Internet, if not just the fabric of the world. They won't need a special label.

But IoT is technology, and technology is not (in our view) an end in itself. What is important to us as designers is how we can use IoT technology to create things that people need, want, and value. Therefore, this book is titled *Designing Connected Products*. We use "products" to refer to both physical things and software services: in IoT, the two are inseparable.

## The Design Challenge

For a long time, much of the discourse around IoT was about hardware hacking, or industrial M2M. The maker movement is empowering people who may not have previously worked with hardware to create their own prototypes with more and more accessible kits, such as Arduino.<sup>7</sup> And grand-scale industrial solutions, such as sensor networks for logistics, agriculture, and environmental monitoring are promising greater efficiency and cost saving for big business.

Somewhere in between those two, the quotidian business of making fully realized, everyday products for consumers felt as if it wasn't really getting the same kind of attention. That's now changed. There are connected thermostats, door locks, drinking cups, fitness monitors, baby sleepsuits, garden sprinklers, catflaps... and on it goes.

There are standout products, but doing good design for IoT is arguably more complex than doing good web service design. Some of this is to do with the current state of the technology. Some of this reflects our as-yet immature understanding of what makes a compelling consumer

---

<sup>7</sup> <http://www.arduino.cc>

IoT value proposition.<sup>8</sup> Some of this is to do with the fact that there are more aspects of design to consider, and tackling them independently creates an incoherent user experience (UX).

IoT is a highly technology-driven field and to date, much of the focus has been around solving connectivity challenges. But as the technology matures, we need to ask ourselves what we want to make from it, and how we should make it. Right now, we often struggle just to imagine what we can use it for. Understanding how to create compelling and usable everyday products and services will be crucial to success—but challenging. This book can't tell you all the answers, but it is intended to help you contribute to the development of the field of consumer IoT design.

## Why Did I Write This Book?

The first aim of this book was to provide a common framework for thinking about design for IoT, setting out the different aspects of the user experience that need to be considered and showing how they fit together.

IoT design is often thought about as a mix of industrial design and app UI. Those are important, but they are only part of the picture. You could create a beautiful app, and a stunning piece of hardware, and users could still have a terrible experience.

I wanted to show that UX is different when interactions are spread across multiple devices. I wanted to show that infusing the real world with the quirks of the network may sometimes feel very weird. I wanted to show that even quite simple connected products are conceptually more complex than nonconnected ones. And I wanted to repeat an obvious point that is far from unique to IoT, but that is often forgotten in the gold rush of a new technology: that you can't make a great design if you don't have a great value proposition.

---

8 Taking a high-level view of the products making waves on crowdfunding sites, you could perhaps view the state of the market as a giant experiment: seeking to understand what can, and should, be connected, as well as how to create value and profit from connecting Things up. It's an inevitable part of this process that some ideas will be stronger than others.

We need a shared understanding of the challenges we may run into, and a common vocabulary for discussing them so that when we use the word “design,” we’re talking about the same things.

The second aim of this book is to introduce the inherent challenges of designing for IoT and give the reader a decent grounding in the technical and human reasons those challenges exist. Some of these technical challenges in particular may come as a surprise to designers accustomed to working with “conventional” Internet devices.

Third, I wanted to write a practical book—the type of book I looked for when I started working in this space, but couldn’t find.

There have been some great theoretical books about connected things in the world. If you haven’t already, I encourage you to read Adam Greenfield’s *Everyware: The Dawning Age of Ubiquitous Computing*<sup>9</sup> and Bruce Sterling’s *Shaping Things*.<sup>10</sup> Both are old in Internet years now, but will make you think deeply and critically about what connected products can and should do—and what they should not do. Mike Kuniavsky’s *Smart Things*<sup>11</sup> was the first practitioner book to bridge theory with design process. Adrian McEwen and Hakim Cassimally’s *Designing the Internet of Things*<sup>12</sup> provides a great primer on creating a hardware product as a maker or startup. But there seemed to be an opportunity to go into depth on the practicalities for user experience design for commercial consumer products.

This is not a cookbook, nor a set of methods. A lot of our current design methods aren’t well suited to the complexity of interconnected systems and they’re going to need to evolve. It’s far too early to set out a process you can follow from end to end that will unfailingly generate a fantastic product. In my view, that’s far too simplistic an approach to design anyway. This book can’t tell you what to do. But we hope it will be able to help you define and solve your own design challenges. We look forward to watching new breakthroughs in design ideas and methods emerge in this field. We hope you will contribute some of them.

---

9 Berkeley, CA: New Riders, 2006.

10 Cambridge, MA: MIT Press, 2006.

11 Burlington, MA: Morgan Kaufman, 2010.

12 John Wiley & Sons, 2013.

As with any technology book, the examples will date quite quickly. But many of the challenges will be with us for some time to come. In this book, we have tried to derive general principles that will stand you in good stead for the time being, even as the technology evolves and new products and platforms appear.

## Why Is It About Consumers?

This is a book about designing products for mass-market consumers.

IoT is too big a space to tackle in one book—on the industrial side, each vertical market alone probably requires its own book. The authors' expertise is primarily in consumer design. But consumers are arguably the most difficult market. They may not be inherently interested in the technology, as we discuss in Chapter 4. They generally have a choice as to whether they use it or not (unlike employees). We are still figuring out most of the business cases for connected consumer products, whereas the benefits of industrial applications (e.g., agriculture and logistics) are often much easier to quantify.

We've included lots of connected home examples because this is arguably the most complex context for consumers. Home is the place where there will be the greatest number of devices to coordinate. Home life is a web of social relationships and daily routines. It's also our refuge from the world: the last place we want to subsume our needs to the demands of unforgiving technology.

That said, there are commonalities in the technology between consumer and industrial applications that will create common challenges in design. Some of the material in this book will also be relevant to designers working on industrial systems, especially those that will be used by people who aren't technical experts.

## Who Is It For?

This book is aimed at two key audiences:

- UX designers/researchers (or UX-oriented product managers) who are fairly new to IoT but have a working knowledge of UX design methods. If that's you, we hope this book will give you a working understanding of the technology insofar as it affects UX, and a solid framework for tackling the design challenges you will face.



- Technologists (or technically oriented product managers) with a working knowledge of connected hardware, embedded software, or web/mobile application development, who want to understand how to make better user experiences. We hope this book will help you understand how technology can shape the UX, and equip you to create technical solutions that support good design.

## About the Authors

Design for IoT is a big topic. It needed the expertise of multiple people to do justice to the breadth and depth we wanted the book to include. This book therefore has multiple authors, each with a different background.

My own background is in digital UX strategy and user research, most recently for connected home applications. Martin Charlier is a rare breed of designer with expertise in product, industrial, interaction, and service design. Liz Goodman wrote the book on user research<sup>13</sup> and has deep expertise in design strategy in networked environments. Alfred Lui has a wealth of experience in designing for health and fitness wearables. Ann Light is a design academic with a long-standing commitment to grasping the social and political impact of mobile and ubiquitous computing. Ann began as my technical editor, helping me detangle my thinking, organize the book and providing expert guidance on content. But after many conversations about the daunting social ramifications of IoT, it was clear she was the best person to write about privacy and ethics.

## How This Book Is Organized

This is a reference book. You can read it cover to cover if you like, or dip in and out of the topics as you need. Each chapter contains cross-references to related topics in other chapters to help you navigate the book.

### *Chapter 1, What's Different About User Experience Design for the Internet of Things?*

by Claire Rowland

This chapter introduces the differences between UX for pure software services, and UX for IoT. It sets out a model of all the different facets of design that are required to deliver a good UX for an IoT

---

<sup>13</sup> Elizabeth Goodman, Mike Kuniavsky, and Andrea Moed, *Observing the User Experience, Second Edition* (Burlington, MA: Morgan Kaufman, 2012).

service. It provides the conceptual foundations for the rest of the book, so whatever your particular area of interest, we recommend that you start with this chapter.

### *Chapter 2, Things: The Technology of Connected Devices*

by Claire Rowland

This chapter explores the technology needed to create the “things” in the Internet of Things: the different types of device and the technology they contain, how sensors and actuators bridge the gap between the physical and digital worlds, and how the challenge of powering IoT devices has a fundamental impact on the technology and UX.

### *Chapter 3, Networks: The Technology of Connectivity*

by Claire Rowland

This chapter explores networking: how the things connect to each other. It covers the impact of networks on UX in IoT, the basics of different types of system architecture and network, network communication patterns, and how API design relates to UX. It’s long and quite detailed, so less technical readers might not want to read it all in one go. But again, these concepts are quite fundamental to understanding some of the unique challenges in designing for IoT.

#### [ NOTE ]

If you’re an IoT engineer, Chapters 2 and 3 will be familiar ground. But you might be interested in the discussion of how the nature of the technology can affect the UX. If you’re new to the technology, we recommend you read at least Chapter 2 and the first part of Chapter 3 (the relevant parts are marked in the chapter). This will help you understand how the nature of the technology creates many of the UX challenges discussed later in the book.

### *Chapter 4, Product/Service Definition and Strategy*

by Claire Rowland

This chapter discusses the challenges of creating great product propositions: a prerequisite to great product and UX design. It considers the differences between tools aimed at innovators and

products for mass-market consumers, what makes a good product, the need to build service offerings around products, and how business models in IoT may take shape.

### *Chapter 5, Understanding People and Context*

by Elizabeth Goodman

This chapter shows how engaging with potential users can help you create successful connected products. It covers concepts and questions that can help guide your understanding of the audience, and methods and tools for answering those questions.

### *Chapter 6, Translating Research into Product Definitions*

by Elizabeth Goodman

This chapter explores how to take what you have learned about your audience, and translate that into a compelling proposition and design strategy. It also shows how to avoid some common pitfalls, when business models, technical capacities, and user expectations may not match up.

### *Chapter 7, Embedded Device Design*

by Martin Charlier

This chapter explores some of the basics of designing physical devices and what's important when designing physical *connected* objects. It's beyond the scope of this book to teach readers to become industrial designers. Rather, we explain the considerations that an industrial designer would take into account when designing a connected device, the process they might follow, and how this is different from software UX or interaction design. The intention is to demystify what industrial designers do, and help forge better collaboration between hardware and software designers.

### *Chapter 8, Interface and Interaction Design*

by Martin Charlier

This chapter explores interface and interaction design for connected devices. Many of these can be used via mobile or web apps, but many also have some kind of onboard user interface. This chapter looks at the challenges of designing interfaces for embedded devices, the pros and cons of different interface types, and best practice examples and recommendations.

## *Chapter 9, Cross-Device Interactions and Interusability*

by Claire Rowland

In systems where functionality and interactions are distributed across more than one device, it's not enough to design individual UIs in isolation. This chapter explores *interusability*—the user experience of interconnected devices and cross-platform interactions—and how to make a bunch of diverse devices feel like they are working in concert.

## *Chapter 10, Interoperability*

by Claire Rowland

The proliferation of different technical standards in IoT means that getting devices to work together is hard. Many devices are locked away in proprietary ecosystems because frequently, that is the easiest way to get them to work. Building on the networking concepts discussed in Chapter 3, this chapter explores the technical challenges of interoperability (and some emerging solutions) and their impact on UX.

## *Chapter 11, Responsible IoT Design*

by Ann Light and Claire Rowland

This chapter looks at the factors that make products trustworthy and safe to use. Connecting up products and services brings new challenges for security, privacy, social engineering, and the environment. We argue that we should consider how our designs impact others' lives, and take this as seriously as we take the need for profit and competitive advantage.

## *Chapter 12, Supporting Key Interactions*

by Claire Rowland with Martin Charlier

There are some key user interactions that are common to different types of connected products. Your product is likely to need some way for users to get the system set up and devices connected, access controls and data, manage devices and alerting behavior, and perhaps configure and manage automated behaviors. This chapter sets out some practical requirements and design approaches to address these needs.

### *Chapter 13, Designing with Data*

by Alfred Lui and Claire Rowland

Networked, embedded devices allow us to capture data from the world that we didn't have before, and use it to deliver better services to users. This chapter considers how we can make sense of the mass of potential available data, and use it to deliver better products and services.

### *Chapter 14, Iterative Design: Prototyping and Learning*

by Elizabeth Goodman

This chapter discusses methods for developing designs for connected products through prototyping and evaluation, including lightweight methods for prototyping initial product ideas, experiences of use, services, interactions with data, and novel types of interaction.

### *Chapter 15, Designing Complex, Interconnected Products and Services*

by Claire Rowland

This chapter looks ahead to a future of complex systems composed of many interconnected devices and applications. It discusses the challenges in making complex systems understandable and valuable, keeping users in control without overwhelming them with configuration options.

## **Acknowledgments**

We are deeply grateful to the awesome editorial team: Mary Treseler for initiating this project and encouraging us throughout, Ann Light (in technical editor capacity) for helping us find order in chaos, and Angela Rufino, Jasmine Kwityn, and Kara Ebrahim for making the book a reality. Thanks also to Helen Codling, Sara Peyton, and Dellaena Maliszewska of O'Reilly marketing for support with events.

Huge thanks to our technical reviewers for their supportive and constructive guidance: Andy Stanford-Clark, Matt Edgar, Jason Mesut, and Ben Bashford. We are particularly grateful to Tom Igoe for his ongoing support with evolving drafts. Pilgrim Beart, Simon Frost, Christopher Osborne, Pamela Pavliscak, and Edd Dumbill also offered invaluable help reviewing specific chapters. Any remaining inaccuracies are the responsibility of the authors.

We also wish to thank those who generously contributed case studies: Arna Ionescu, Matt Edgar, Parrish Hanna, Jeff Maina, and Adam Scheuring.

We are indebted to the many others who helped this book come about. Some generously donated their time for interviews and discussions, pointed us to research or helped us access other useful contacts. Some tested products and helped source images. Some are colleagues or collaborators whose insights helped shape the ideas that coalesced into this book. They are: Helen Le Voi, Gawain Edwards, Pertti Huuskonen, Alex von Feldmann, Martin Storey, Naintara Land, Mike Kuniavsky, Scott Jenson, Katarina Segerstahl, Kaisa Väänänen-Vainio-Mattila, Jack Schulze, Denise Wilton, Alexandra Deschamps-Sonsino, Louisa Heinrich, Chris Browne, Patrick Bergel, Adrian McEwen, Whan Stransom, Usman Haque, Martin Spindler, Alan Blackwell, Saeed Aghae, Simon Johnson, Giles Colborne, Paul Connell, Carson Darling, Toby Jaffey, Anna Kuriakose, Ben Stewart, Colin Chapman, Andy Hobsbawm, Niall Murphy, Ben Reason, Michael Koster, Anders Mellbratt, Chris Palmer, Adrienne Porter Felt, Gemma Brady, Jody Haskayne, Thomas Amberg, Jordan Husney, Niel Bornstein, Tun Shwe, Fei Phoon, Ji-Hye Park, Katja Egmoose, Melanie Wendland, Hsi-Pei Liao, Nicola Combe, Kassir Hussain, Vamshi Lingampally, Jacqui Brown, Sarah Novotny, Catherine Mayer, William Gibson, Ian Brown, Karten Design, Alex Harding, David Merrill, Deena Rosen, Juliana Rotich, Jon Bruner, Venkatesh Prasad, Barry Fischer, Boris Adryan, Timo Arnall, Paul Backett, Durrell Bishop, Andy Budd, Jill Butler, Matt Cockerill, Sam Crosland, Sabine Croxford, Anna Jones, Freya Dobrindt, Jared Ficklin, Andy Huntington, Victor Johansson, Devraj Joshi, Lucy Kirby, William Lidwell, Tom Metcalfe, Andrew Nicolaou, Eduardo Aguilar Pelaez, Patrik Schmidberger, Matt Shannon, Olivia Solon, Denise Stephens, Matt Webb, Craig Wightman, Stuart Wood, Max Gadney, and Ljuba Miljkovic. Sincere apologies to anyone we have inadvertently omitted from this list.

## **PERSONAL ACKNOWLEDGMENTS**

Claire Rowland: I could not have written this without the extraordinary support of my partner Simon, mum Christine, my parents-in-law Janice and Allan, and Kathy. Thank you, you rock.

Martin Charlier: For their ongoing support during this project, I'd like to thank my partner Freya, my family, and my friends.

Alfred Lui: I want to thank my wife Sonja, my mother Lelia, and my brother Clarence for their love, patience, and support during this project.

# What's Different About User Experience Design for the Internet of Things?

**BY CLAIRE ROWLAND**

User experience (UX) design and human–computer interaction (HCI) emerged in a world of desktop computers. But our experience of computing has changed radically in the past 10–15 years. Many of our interactions now take place on mobile phones, tablets, ereaders, and smart TVs. And it's common to use one service across multiple devices with different form factors (Figure 1-1).



**FIGURE 1-1**

BBC iPlayer can be used on connected TVs, smartphones, tablets, PCs, game consoles, and set-top boxes (image: BBC)

We're still figuring out the best ways to design for new devices and experiences. Interactions can happen in a wide variety of contexts, especially for mobile devices. They can happen on a variety of scales,

from tiny wrist-tops, to smartphones, to TV user interfaces (UIs) viewed from 10 feet away. Even academic researchers in HCI have published relatively few papers on cross-platform design.

The “Internet of Things” (IoT) refers to the growing range of everyday objects acquiring connectivity, sensing abilities, and increased computing power. In consumer terms, some common categories currently include:

- Connected home technology (such as thermostats, lighting, and energy monitoring)
- Wearables (such as activity/fitness trackers and “smart” watches)
- Medical/wellness devices (such as bathroom scales and blood pressure monitors)
- Connected cars (which may provide access to smartphone apps via dashboard controls, engine diagnostics, and automatic alerting of authorities in case of a crash)
- Urban systems (such as air quality sensors, city rental bikes, and parking meters/sensors)

Designing for the Internet of Things (IoT) raises all the challenges of cross-platform design, and more.

An obvious difference is the much wider variety of device form factors, many without screens (see Figure 1-2).

Less obvious differences include the effects of many IoT devices being only intermittently connected. And even a simple task, like unlocking a door, can quickly become complex when it forms part of a system spanning many interconnected devices, services, and users.

IoT is still a technically driven field. At the time of writing, the UX of many IoT products is some way off the level expected of mature consumer products. For example, the UK government commissioned a study on the usability of connected heating systems in late 2013. They found that none of the five major connected heating devices on the market in the UK offered a good UX.<sup>1</sup>

---

<sup>1</sup> “Usability Testing of Smarter Heating Controls,” Amberlight Partners for the Department of Energy and Climate Change, <http://bit.ly/195jGLu>.





**FIGURE 1-2**

The Lockitron connected door lock is one of a huge number of connected devices with no screen (image: Lockitron)

Before we start, we should explain what we mean by “UX” and “user experience design.” Many people equate the term with “UI” or “user interface design,” but they are not the same. UX is a holistic term referring to a wide range of design disciplines involved in creating systems that are useful, usable, and pleasurable to use. UI design is just one of those. As the UX consultant Elisabeth Hubert explains it:

The user interface is not the (design) solution, but instead is the medium through which users interact with the solution.<sup>2</sup>

(In “A Design Model for IoT,” we’ll look in more detail at the facets of design that comprise UX for an IoT system.)

In this chapter, we begin by introducing the differentiators that make UX design for IoT a new and challenging domain.

This chapter introduces:

- What’s different about UX for IoT (see page 4)
- A design model for IoT (see page 17)

---

2 “Interaction Design Beyond the Interface,” <http://bit.ly/195ktvO>.

It considers the following issues:

- The challenges of distributing functionality across multiple devices (see page 5)
- How the focus of the UX is increasingly in the service (see page 6)
- Whether we are ready for the real world to start behaving like the Internet (see page 7)
- How the ways devices connect to the network affects the UX (see page 8)
- How multiple devices create more complexity for the user to understand (see page 9)
- How controlling distributed devices is similar to programming (see page 11)
- How what seem like simple systems can rapidly become complex (see page 13)
- The problems of having many different technical standards (see page 14)
- How data is at the core of many IoT services (see page 16)
- The layers of UX thinking required to create a successful IoT product: from UI and interaction design all the way down to the platform (see page 17)

## How Is UX for IoT Different?

Designing for IoT comes with a bunch of challenges that will be new to designers accustomed to pure digital services. How tricky these challenges prove will depend on:

- The maturity of the technology you're working with
- The context of use or expectations your users have of the system
- The complexity of your service (e.g., how many devices the user has to interact with)

The following sections summarize the key differences between UX for IoT and UX for digital services. Some of these are a direct result of the technology of embedded devices and networking. We'll explain the technology issues in more detail in Chapters 2 and 3. But even if you are already familiar with embedded device and networking technology, you might not have considered the way it shapes the UX.

### **FUNCTIONALITY CAN BE DISTRIBUTED ACROSS MULTIPLE DEVICES WITH DIFFERENT CAPABILITIES**

IoT devices come in a wide variety of form factors with varying input and output capabilities. Some may have screens, such as heating controllers or washing machines (see Figure 1-3). Some may have other ways of communicating with us, such as flashing LEDs or sounds (see Figure 1-4).



**FIGURE 1-3**

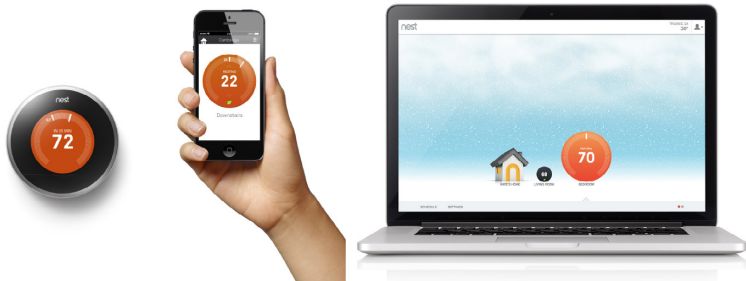
The Honeywell evohome connected radiator valve has a basic LCD screen (image: Honeywell Environmental Controls)

Some may have no input or output capabilities at all and are unable to tell us directly what they are doing. Interactions may be handled by web or smartphone apps. Despite the differences in form factors, users need to feel as if they are using a coherent service rather than a set of disjointed UIs. It's important to consider not just the usability of individual UIs but *interusability*: distributed user experience across multiple devices (see Figure 1-5). This is explained further in Chapter 9.



**FIGURE 1-4**

The GlowCaps connected pill bottle lid uses light and sound notifications to remind the user to take medication (image: GlowCaps)



**FIGURE 1-5**

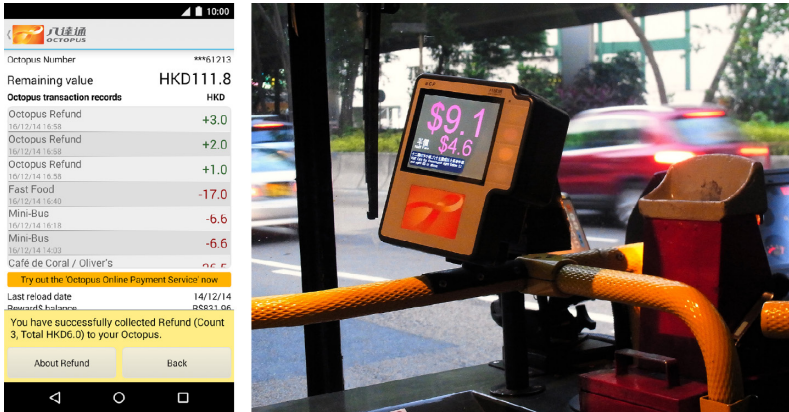
The Nest Learning Thermostat can be controlled by the on-device UI, a smartphone app, or a web app (image: Nest)

## THE FOCUS OF THE USER EXPERIENCE MAY BE IN THE SERVICE

When we talk about IoT, we tend to focus on the devices, particularly those with striking or novel forms. But the behavior of the device might be generated by a program that lives on another device on the network (i.e., a server). We call this the Internet (or “cloud”) service.

This means that the service around a connected device is often just as critical in delivering the user experience, if not more so, than the device itself. For example, the smart travelcards such as the London Oyster and Hong Kong Octopus are often thought of as the focus of the payment service. But the services can be used without a card at all via

an NFC-enabled smartphone or bank card (Figure 1-6). The card is just an “avatar” for the service (to borrow a phrase from the UX expert Mike Kuniavsky).<sup>3</sup> For more on service business models and product–service ecosystems, see Chapter 4.



**FIGURE 1-6**

Hong Kong's Octopus payment service can be used with an NFC phone as well as a smart card (Octopus reader on KMB bus by Ka890, CC licence via Wikimedia Commons; Octopus app: Octopus Cards Ltd.)

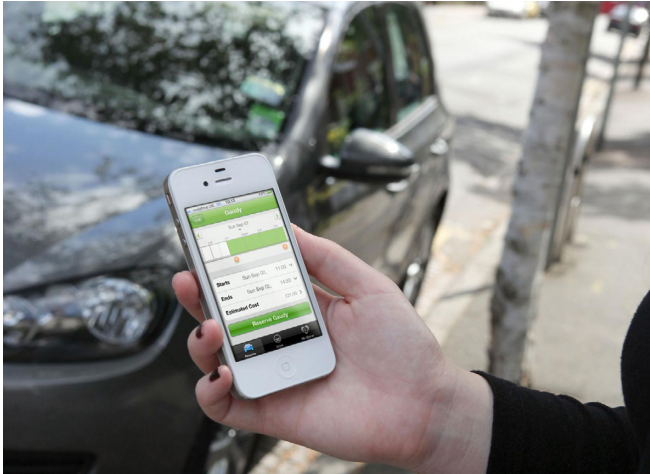
## **WE DON'T EXPECT INTERNET-LIKE GLITCHES FROM THE REAL WORLD**

It's frustrating when a web page is slow to download or a Skype call fails. But we accept that these irritations are just part of using the Internet. By contrast, real-world objects respond to us immediately and reliably.

When we interact with a physical device over the Internet, that interaction is subject to the same latency and reliability issues as any other Internet communication. So there's the potential for delays in response and for our requests and commands to go missing altogether. This could make the real world start to feel very broken. Imagine if you turned your lights on and they took two minutes to respond, or failed to come on at all.

3 Mike Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design* (Burlington, MA: Morgan Kaufmann, 2010).

In theory, there could be other unexpected consequences of things adopting Internet-like behaviors. In the Warren Ellis story “The Lich House”<sup>4</sup> a woman is unable to shoot an intruder in her home: her gun cannot contact the Internet for the authentication that would allow her to fire it. This might seem far-fetched, but we already have objects that require authentication, such as Zipcars (Figure 1-7).



**FIGURE 1-7**

When you book a Zipcar online, the service sends details of the reservation to the car; swiping a smart card authenticates you as the person who made the booking (image: Zipcar)

For more information on networking and its impact on UX, see Chapter 3.

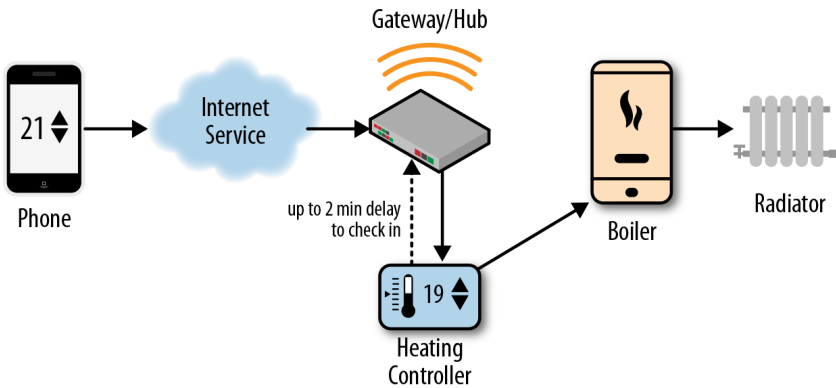
## **IOT IS LARGELY ASYNCHRONOUS**

When we design for desktops, mobiles, and tablets, we tend to assume that they will have constant connectivity. Well-designed mobile apps handle network outages gracefully, but tend to treat them as exceptions to normal functioning. We assume that the flow of interactions will be reasonably smooth, even across devices. If we make a change on one device (such as deleting an email), it will quickly propagate across any other devices we use with the same service.

---

<sup>4</sup> “Lich House,” <http://www.iftf.org/fanfutures/ellis/>.

That will not always happen in IoT systems. Many connected devices run on batteries, and need to conserve electricity. Maintaining network connections uses a lot of power, so they only connect intermittently. This means that parts of the system can be out of sync with one another, creating discontinuities in the user experience. For example, if your heating is set to 19° C, and you use the heating app on your phone to turn it up to 21° C, it will take a couple of minutes for your battery-powered heating controller to go online to check for new instructions. During this time, the phone says 21° C, and the controller says 19° C (Figure 1-8).



**FIGURE 1-8**

Schematic of heating system with app and controller giving different status information

These discontinuities won't always be noticed: sometimes the delays will be very short, and sometimes users won't be around to notice them—for example, when they are turning on a remote device. So the UX may *feel* synchronous, even if the service technically isn't. But when we do notice, the UX may feel quite disjointed.

For more on the technical background see Chapters 2 and 3. For more on the design impact, see Chapter 9.

### CODE CAN RUN IN MANY MORE PLACES

The configuration of devices and code that makes a system work is called the *system model*. In an ideal world, users should not have to care about this. We don't need to understand the technical architecture of a conventional Internet service, like Amazon, in order to use it

successfully. Instead, we form a *conceptual model* of what Amazon does and how it works that's good enough to help us understand what to do. We know we can search or browse products, that we need to add them to a basket, set up or log into a user account, pay, and then we'll get a delivery. We don't need to understand the different machines involved in making this system function.

But as a consumer of an IoT service right now, you can't always get away from some of this technical detail.

A typical IoT service is composed of:

- One or more *embedded* devices (the *things* in IoT, described in Chapter 2)
- An Internet service
- Perhaps a *gateway* device (a separate device needed to connect some embedded devices to the Internet, described in Chapter 3)
- One or more mobile or web apps for the user to interact with the service via a mobile, tablet, or PC

Compared to a conventional web service, there are more places where code can run. There are more parts of the system that can, at any point, be offline. Depending on what code is running on which device, some functionality may at any point be unavailable.

For example, imagine you have a connected lighting system in your home. It has controllable bulbs or fittings, perhaps a gateway that these connect to, an Internet service, and a smartphone app to control them all (see Figure 1-9). You have an automated rule set up to turn on some of your lights at dusk if there's no one home. If your home Internet connection goes down, does that rule still work? If the rule runs on the Internet service or your smartphone, it won't. If it runs on the gateway, it will. As a user, you want to know whether your security lights are running or not. You need to understand a little about the system model to understand which devices are responsible for which functionality, and how the system may fail.

It would be nice if we could guarantee no devices would ever lose connectivity, but that's not realistic. And IoT is not yet a mature set of technologies in the way that ecommerce is, so failures are likely to be more frequent. System designers have to ensure that important functions



(such as home security alarms) continue to work as well as possible when parts go offline and make it as easy as possible for users to understand what’s happening, and recover from any problems.

For more on matching system models to users’ conceptual models, see Chapter 6.



**FIGURE 1-9**

The Philips Hue system consists of connected bulbs, a gateway, an Internet service, and a smartphone app (image: Philips)

## **DEVICES ARE DISTRIBUTED IN THE REAL WORLD**

The shift from desktop to mobile computing means that we now use computers in a wide variety of situations. Hence, mobile design requires a far greater emphasis on understanding the user’s needs in a particular *context of use*. IoT pushes this even further: computing power and networking is embedded in more and more of the objects and environments around us. For example, a connected security system can track not just whether the home is occupied, but who is inside, and potentially video record them. Hence, the social and physical contexts in which connected devices and services can be used are even more complex and varied.

## **REMOTE CONTROL AND AUTOMATION ARE PROGRAMMING-LIKE ACTIVITIES**

In 1982, the HCI researcher Ben Shneiderman defined the concept of *direct manipulation*. User interfaces based on direct manipulation “depend on visual representation of the objects and actions of interest,

physical actions or pointing instead of complex syntax, and rapid incremental reversible operations whose effect on the object of interest is immediately visible. This strategy can lead to user interfaces that are comprehensible, predictable and controllable.”<sup>5</sup> Ever since, this has been the prevailing trend in consumer UX design (see Figure 1-10). Direct manipulation is successful because interface actions are aligned with the user’s understanding of the task. They receive immediate feedback on the consequences of their actions, which can be undone.



**FIGURE 1-10**

An image by the designer Susan Kare, created in MacPaint 1.0: an early example of a popular direct manipulation interface (image: Wikipedia)

IoT creates the potential for interactions that are displaced in time and space: configuring things to happen in the future, or remotely. For example, you might set up a home automation rule to turn on a video camera and raise the alarm when the house is unoccupied and a motion sensor is disturbed. Or you might unlock your porch door from your work computer to allow a courier to drop off a parcel.

Both of these break the principles of direct manipulation. To control things that happen in the future, you must anticipate your future needs and abstract the desired behavior into a set of logical conditions and

5 Ben Shneiderman, “Direct Manipulation for Comprehensible, Predictable and Controllable User Interfaces,” *Proceedings of the ACM International Workshop on Intelligent User Interfaces* 1997: 33–39.

actions. As the HCI researcher Alan Blackwell points out, this is basically programming.<sup>6</sup> It is a much harder cognitive task than a simple, direct interaction. That's not necessarily a bad thing, but it may not be appropriate for all users or all situations. It impacts usability and accessibility.

Unlocking the door remotely is an easier action to comprehend. But we are distanced from the consequences of our actions, and this poses other challenges. Can we be sure the door was locked again once the parcel had been left? A good system should send a confirmation, but if our smartphone (or the lock) lost connectivity, we might not receive this.

For a deeper discussion of programming-like experiences, see Chapter 15.

### **COMPLEX SERVICES CAN HAVE MANY USERS, MULTIPLE UIs, MANY DEVICES, MANY RULES AND APPLICATIONS**

A simple IoT service might serve only one or two devices (e.g., a couple of connected lights). You could control these with a very simple app. But as you add more devices, there are more ways for them to coordinate with one another. If you add a security system with motion sensors and a camera, you may wish to turn on one of your lights when the alarm goes off. So the light effectively belongs to two functions or services: security and lighting. Then add in a connected heating system that uses information from the security system to know when the house is empty. And assume that there are several people in the house with slightly different access privileges to each system. For example, some can change the heating schedule, some can only adjust the current temperature. Some have admin rights to the security system, some can only set and unset the alarm. What started out as a straightforward system has become a complex web of interrelationships.<sup>7</sup>

---

6 In conversation. See Alan F. Blackwell, "What Is Programming?" *Proceedings of PPIG* 2002: 204–218.

7 Access control is a feature of many home automation systems. It has not previously been seen as a need for most household appliances or systems, aside from home security. Some people might argue that it still mostly isn't necessary, and that it's a result of an overly technical mindset that treats homes like computer operating systems (in which everyone is an identified user, with an account and access privileges). In Chapters 5 and 6, we discuss issues such as this in the context of appropriate design for user needs and social context. But for systems that do go down this route, it does add significant complexity.

For a user, understanding how this system works will become more challenging as more devices and services are added. It will also become more time consuming to manage.

For more information, see Chapter 15.

## **MANY DIFFERING TECHNICAL STANDARDS MAKE INTEROPERABILITY HARD**

The Internet is an amazing feat of open operating standards, but, before embedded devices were connected, there was no need for appliance manufacturers to share common standards. As we begin to connect these devices together, this lack of common technology standards is causing headaches. Just getting devices talking to one another is a big enough challenge, as there are many different network standards. Being able to get them to coordinate in sensible ways is vastly more complicated still.

The consumer experience right now is of a selection of mostly closed, manufacturer-specific ecosystems. Devices within the same manufacturer's ecosystem, such as Withings (see Figure 1-11), will work together. But this is the only given. In the case of Withings, this means that devices share data with a common Internet service, which the user accesses via a smartphone app. Apple's Airplay is an example of a proprietary ecosystem in which devices talk directly to one another.

We're starting to see manufacturers collaborating with other manufacturers too. So your Nest Protect smoke detector can tell your LIFX lightbulbs to flash red when smoke is detected. (This is done by connecting the two manufacturer's Internet services rather than connecting the devices.)

There are also some emerging platforms that seek to aggregate devices from a number of manufacturers and enable them to interoperate. The connected home platform SmartThings supports a range of network types and devices from manufacturers such as Schlage and Kwikset (door locks); GE and Honeywell (lighting and power sockets); Sonos (home audio); and Philips Hue, Belkin, and Withings (connected home products); see Figure 1-12. But the platform has been specifically configured to work with each of these. You cannot yet buy any device and assume it will work well with SmartThings.



**FIGURE 1-11**  
The Withings ecosystem of devices (images: Withings)



**FIGURE 1-12**  
The SmartThings gateway and some compatible devices (image: SmartThings)

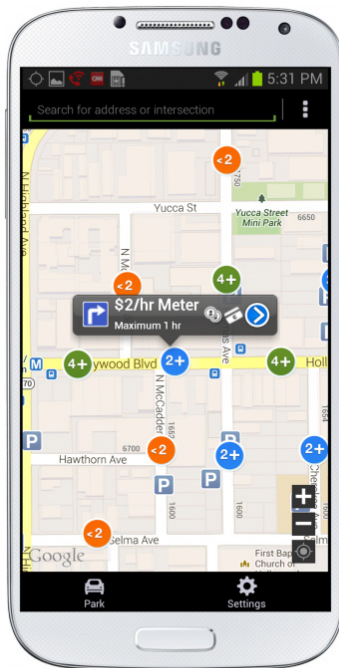
For the near future, the onus will be largely on the consumer to research which devices work with their existing devices before purchasing them. Options may be limited. In addition, aggregating different types of device across different types of network tends to result in a lowest common denominator set of basic features. The service that promises to unify all your connected devices may not support some of

their more advanced or unique functions: you might be able to turn all the lights on and off but only dim some of them, for example. It will be a while before consumers can trust that things will work together with minimal hassle.

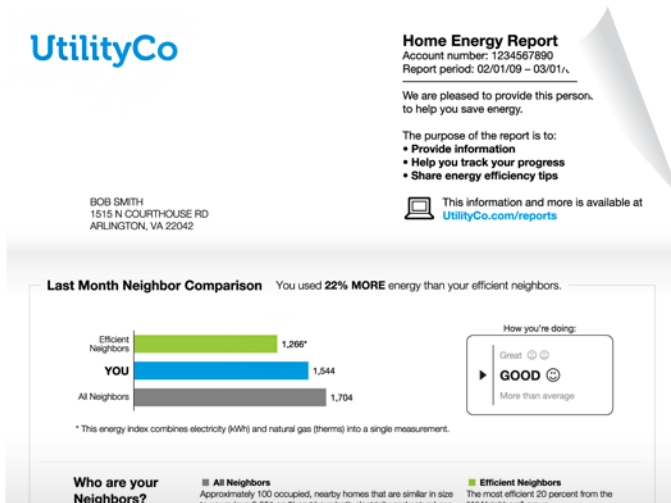
For more information, see Chapter 15.

## IOT IS ALL ABOUT DATA

Networked, embedded devices allow us to capture data from the world that we didn't have before, and use it to deliver better services to users. For example, drivers looking for parking spaces cause an estimated 30% of traffic congestion in US cities. Smart parking applications such as Streetline's Parker use sensors in parking spaces to track where spaces are free, for drivers to find via a mobile app (see Figure 1-13). The software company Opower analyzes data from smart meters to suggest ways in which utility customers could save energy and money (see Figure 1-14).



**FIGURE 1-13**  
Streetline's Parker app (image: Streetline)



**FIGURE 1-14**  
 Sample Opower energy report (image: Opower)

Networked devices with onboard computation are also able to use data, and in some cases act on it autonomously. For example, a smart energy meter can easily detect when electrical activity is being used above baseload. This is a good indicator that someone is in the house and up and about. This data could be used by a heating system to adjust the temperature or schedule timing.

To paraphrase another quote from Mike Kuniavsky, “information is now a design material.”<sup>8</sup>

For more information, see Chapter 13.

## A Design Model for IoT

As shown in the preceding sections of this chapter, designing for IoT will confront you with some extra challenges and complexity that you wouldn’t encounter on a “conventional” (software only) web service. You’ll need to think about some different and perhaps new areas of design that all serve to shape the UX.

<sup>8</sup> Kuniavsky, *Smart Things*, 43

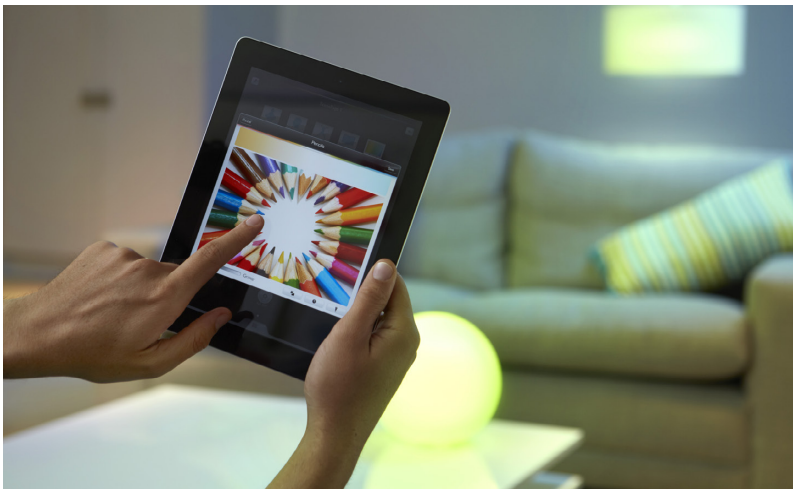
The two most visible and tangible forms of design for IoT are:

*The UI/visual design*

For example, the screen layout and look and feel of the web or mobile apps, or devices themselves (see Figure 1-15). (UIs don't have to be visual; they can use audio, haptics, and other channels, as discussed in Chapter 8. But it's rare for a service to have no screen-based UI at all.)

*The industrial design of the physical hardware*

The form factor, styling, and capabilities of the connected devices themselves (see Figure 1-16).



**FIGURE 1-15**

The Philips Hue UI allows users to change the color of light emitted by an LED bulb (image: Philips Communications)

UI and industrial design are important but not the whole picture. The UX is not just shaped by what the user can see or encounter directly. To create a valuable, appealing, usable, and coherent IoT service, we have to consider design on many different layers.





**FIGURE 1-16**

The Nest Learning Thermostat has a striking industrial design (image: Nest)

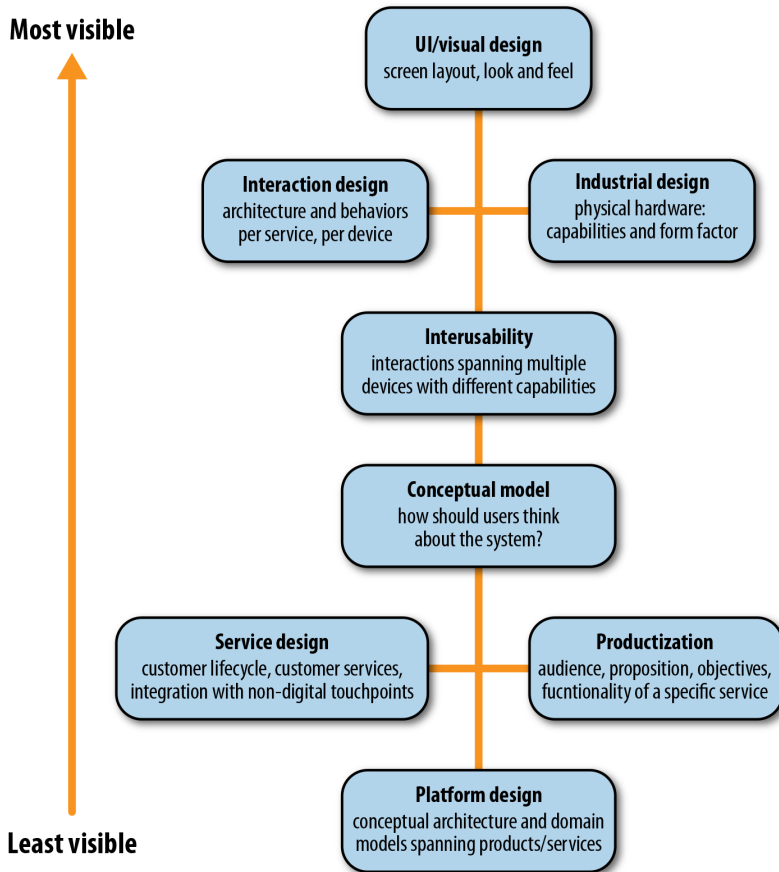
In 2000, Jesse James Garrett produced his “Elements of User Experience” diagram (and subsequent book) to explain how different design specialties fit together in web UX design.<sup>9</sup> This represented the different types of design required, where uppermost layers (i.e., visual design, information, interface, and navigation design) are most visible to the user, but depend on the structure provided by the lower layers (i.e., site objectives, content requirements, etc.), which are dealt with earlier in the project. Just as there were dependencies in his model, where work that was not directly apparent to the user determined aspects that they could directly experience, so designing for connected products has a similar flow, with critical decisions that affect UX made in early stages of the design.

If you’re more familiar with engineering models, you might think of this as being a little like a technology stack (as in the Internet network stack in Chapter 3), in which each layer of technology is dependent on the lower levels functioning well. It’s not an exact analogy, as the dependencies between different layers of design are much more fluid. But it’s a useful comparison to make the point that good design at the higher, more visible layers requires a clear framework at the lower layers.

---

<sup>9</sup> Jesse James Garrett, *The Elements of User Experience: User-Centered Design for the Web and Beyond*, 2nd Edition (Berkeley, CA: New Riders, 2010).

There are many different facets of design involved in delivering a good UX for an IoT service, set out in Figure 1-17.



**FIGURE 1-17**

Facets of design in IoT—a good product requires integrated thinking across all of these

This isn't a set of discrete activities required in your project plan. It says nothing of user research, for example. Nor is it a set of job roles you need on your project. You might, for example, need data analytics, and you'll certainly need engineers.

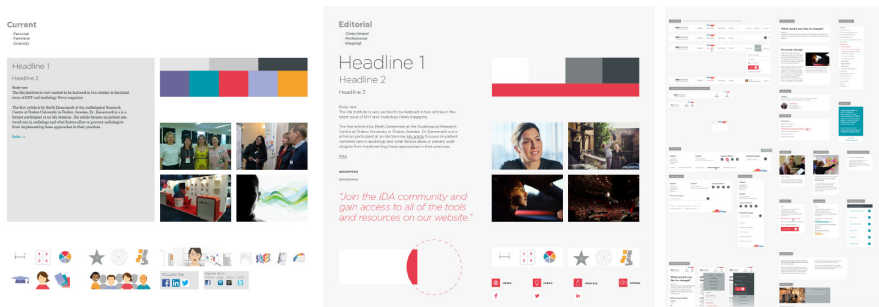
It's aspects of the user experience that need to be considered. Some of these things will evolve in tandem. For example, UI, interaction design, and interusability need to be thought about together. UX design at the platform layer will emerge as a need once you start adding multiple devices to a service.

A good overall product requires integrated thinking across all these layers. A stunning UI means nothing if your product concept makes no sense. A beautiful industrial design may sell products in the short term but can't mask terrible service.

Depending on the type and complexity of your service, layers will require more or less of your time. IoT services aspire to extend over more devices and become more complex over time, and the parts you need less now may become more relevant to you in the future.

## UI/VISUAL DESIGN

UI/visual design refers to screen layout, visual styling, and look and feel on a device. This is the *form* that a device interface takes. The outputs of UI/visual design are typically high-fidelity screen mockups (Figure 1-18). Not all UIs are visual, of course: for a gestural or audio interface, the equivalent function might be defining the aesthetics of the gestures or voice.



**FIGURE 1-18** Style tiles (visual concepts) and UI modules for a redesign of the IDA Institute's website ([idainstitute.com](http://idainstitute.com); images: Halei Liu and Hans Pelle Jart for [Kontrapunkt.com](http://Kontrapunkt.com))

See Chapter 8 for more details on designing UIs for embedded devices.

## INTERACTION DESIGN

Interaction design is the design of device behaviors. Interaction designers shape the sequences of actions between the user and the device needed to achieve particular goals or activity. They also determine how to organize the user-facing functions of the device. For example, a heating controller might have several modes, such as schedule on/off or frost protection, and some hierarchical functions, such as schedule setting. The organization of these functions defines how easy or otherwise it may be for users to find their way around them.

Interaction design is closely aligned to UI design in the sense that the two are usually done in tandem and often by the same people. But interaction design is primarily concerned with behaviors and actions, whereas UI/visual design is concerned with layout and aesthetics. (Just to confuse matters, some people use UI design as a shorthand term to include both interaction design and visual design.) Typical outputs for interaction design might include user flows, low-medium fidelity interactive prototypes, and for a visual UI, screen wireframes (Figure 1-19).



**FIGURE 1-19**

Device and app interaction design concepts for Hackaball, a programmable ball for children ([www.hackaball.com](http://www.hackaball.com); images: Map and Made by Many)

You sometimes hear the term “information architecture” used to describe organization schemes for functionality, but technically this refers to the equivalent activity for content-driven systems, such as content-based websites.

See Chapter 8 for more details.

## **INTERUSABILITY**

Interusability is a relatively new term. It refers to the additional considerations of designing interactions that span multiple devices. The goal is to make the overall experience feel like a coherent service, even when the devices involved may have quite different form factors and input/output capabilities.

Interusability isn’t a separate set of design activities. It’s an extra set of considerations to be addressed in tandem with interaction and UI design. The key differences to a single device UX design process would typically be:

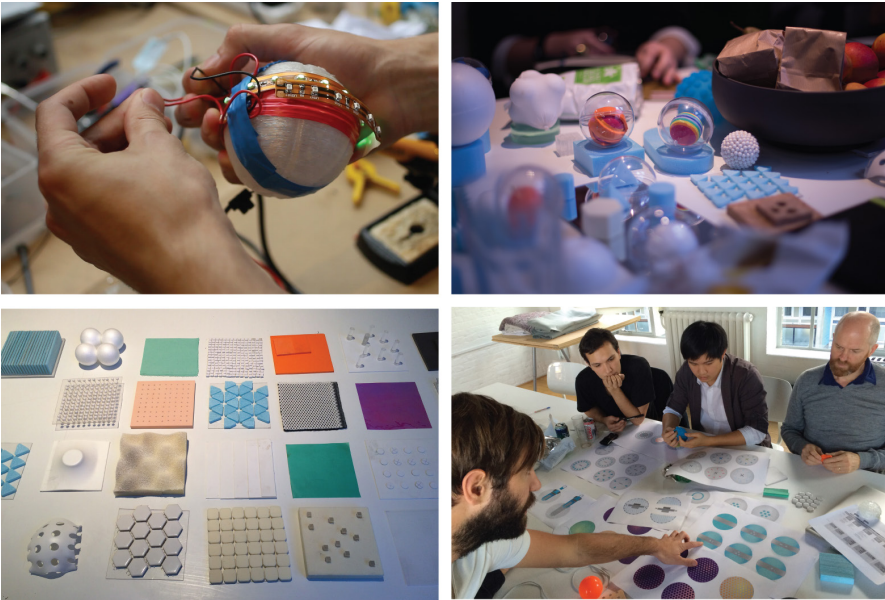
- Specifying which functionality belongs on each device
- Creating design guidelines that span multiple device types
- Designing cross-device user flows for key interactions
- Designing multiple device UIs in parallel

See Chapter 9 for more details.

## **INDUSTRIAL DESIGN**

Industrial design refers to the aesthetic and functional design of the physical hardware in the service: the choice of form, materials, and capabilities it may have (see Figure 1-20). Connected devices contain electronic circuitry and radio antennae, which impose particular requirements on the industrial design. Devices can also have input and output capabilities, which require collaboration between industrial designers and UI/interaction design/interusability.

See Chapter 7 for more details.



**FIGURE 1-20**

Hardware prototype and color/material/form explorations for Hackaball, a programmable ball for children ([www.hackaball.com](http://www.hackaball.com); images: Map and Made by Many)

## SERVICE DESIGN

A connected device is rarely a one-off purchase. It comes with the expectation of an ongoing service, at the very least the provision of the Internet service that keeps it running, and customer support. All of this forms part of the user's overall experience with the product.

Service design is an emerging discipline that addresses this holistic view of user experience. It looks at the whole lifespan of a user's experience with a service, provides a view of all the components of the user experience, and specifies how these function together as a coherent whole (see Figure 1-21).

In addition to device interactions, it might include:

- Customer support interactions
- Instructional guides
- Marketing or sales materials

- In-store experiences
- Email communications and notifications
- The UX of software updates and rolling out new functionality



**FIGURE 1-21**

Experience mapping and prototyping service interactions for obstetric services in Kenya, Uganda, and India (images: M4ID)

The service experience is critical to the success of an IoT product, but in this book, we have not addressed service design as a separate activity or chapter. We encourage you to take a holistic approach to designing the user experience that includes interactions not just with devices but also the wider service context. Service design methods may be extremely useful here, but you would not apply them differently on an IoT project than you would on any other project, so we have not discussed them separately.

See Chapters 4 and 6 for guidance on scoping the service experience. The suggested methods in Chapter 14 include guidance on designing the service experience as part of overall UX design.

## **CONCEPTUAL MODEL**

The conceptual model is the understanding and expectations you want the user to have of the system. What components does it have, how does it work, and how can they interact with it? It's the mental scaffolding that enables users to figure out how to interact with your service. Regardless of whether a conceptual model is designed or not, users will form one. If they get it wrong, they'll struggle to use the system. IoT services are often inherently complex systems. You can create a clear conceptual model through careful system and interaction design and supporting documentation. You want users to feel in control from the start—they should feel confident that they will be able to use the system, even if they don't understand all the details yet.

See Chapters 6 and 9 for more details.

## **PRODUCTIZATION**

Productization is the activity of defining a compelling product proposition. It addresses the audience, proposition, objectives, and overall functionality of service (and often its business model). Does your product solve a real problem for a real audience? Is it presented so that they understand that? Does it appeal to them? This isn't always the domain of the UX designer on a project, but it's the underpinnings of good UX. All the frontend design in the world won't make a killer product unless it does something of value for people, in a way that appeals to them and makes sense.

See Chapter 4 for more details.

## **PLATFORM DESIGN**

A platform is a software framework. It takes care of low-level details to help developers build applications more easily. At their most basic, IoT platforms make it easy to put data from connected devices onto the Internet. Slightly more advanced platforms may provide frameworks to enable different types of devices to interoperate.

A software platform will aim to solve many technical issues, many of which may not directly have an impact on the UX. But some more advanced platform functionality is very much to do with UX.



For example, a platform like Hue or Withings may provide standard ways to:

- Discover new devices and applications
- Add devices and applications onto the system
- Manage devices and users
- Manage how devices share data

These are basic building blocks for the UX. If they don't work well for your users, your UI and interaction design will be full of awkward workarounds.

A more complex platform might also provide ways of organizing and coordinating multiple devices. For example, if a user adds a light to an existing home system, they might reasonably expect the system to know that it should control it along with their other lights and/or offer it as part of the security system. It's not important to make it talk to the toaster. That may be common sense to a human. But the system won't know that unless this kind of logic is encoded in the platform.

If you're building a very simple system of a single device, you might not need to do much platform-level UX to start with. For example, you will need to consider how the user gets the device onto the network. But you don't necessarily need to design a standardized approach that would work for other types of device as well. But once your system has multiple, interconnected devices, there will be design challenges that will require platform logic to solve. Designers should get involved in shaping platforms to ensure they support good higher-level UX.

There is no commonly understood set of activities for this yet. Chapter 15 discusses what these challenges may look like and some of the approaches we might take to solving them.

## Summary

UX for connected devices is not just about UI and interaction design. It also requires designers to think about interusability, industrial design, service design, conceptual models, productization, and platform design.

In summary, it differs from “conventional” UX in the following ways:

Embedded devices often save power by connecting only intermittently	... which means parts of the system can be out of sync, creating discontinuities in UX
Latency on the Internet is out of your control (and reliability is not 100%)	... which means although we expect physical things to respond immediately and reliably, this might not happen
Code can run in many more places	... which means users have to engage with the system model to predict how it will work if parts are offline
Devices are distributed in the real world	... which means social and physical context of use is complex and varied
Functionality can be distributed across multiple UIs	... which means designers need to consider not just usability but interusability
Much of the information processing happens in the Internet service	... which means the service experience is often equally or more important than the single device UX
Remote control and automation are programming-like activities	... which means IoT breaks direct manipulation—the basis of most successful consumer UXes
Many differing technical standards	... which means getting things to work together is hard
Complex services can have many users, many UIs, many devices, many rules and applications	... which means understanding and managing how they all interrelate can be extremely difficult. Users will turn off if admin becomes too onerous.
IoT enables us to capture and act on data we didn't have before	... which means designers need to understand how to use information as a design material

# Things: The Technology of Connected Devices

**BY CLAIRE ROWLAND**

This chapter looks at the technology needed to create the “things” in the Internet of Things (IoT). IoT is creating an explosion in the diversity of devices connected to the Internet. We’re seeing familiar objects gain connectivity and increased computational power, as well as new categories of device that can only exist as a result of the network. Sensors and actuators create new possibilities for bridging information and actions in the real and digital worlds. But many of these devices will have to make careful use of energy and computing power. These technical constraints are the root cause of some design challenges that UX designers will encounter when working with connected devices.

This chapter introduces:

- The different types of IoT device, and the technology they contain (see page 30)
- How sensors and actuators bridge the physical and digital worlds (see page 48)
- The challenge of powering IoT devices and how this shapes the technology (see page 53)

This chapter addresses the following issues:

- How the computer hardware and software technology inside specialized embedded devices—the novel devices of the Internet of Things—differs from multipurpose computers (see page 32)
- How objects with no onboard computing power can nonetheless have an Internet presence (see page 42)

- How sensors can be used to gather data from the physical world (see page 48)
- How actuators can be used to translate digital commands into real-world actions (see page 52)
- Why energy conservation is at the root of many UX design constraints in IoT (see page 53)

### [ NOTE ]

The focus in this chapter is on the *edge devices*—the end points through which the digital service interacts with the real world. There are some other types of IoT device, such as Internet gateways, which are there to provide network infrastructure so that the edge devices can connect to the Internet. See Chapter 3 for more information on gateways.

## Types of Connected Device

Connected devices can take a number of forms, and are typically one of the following:<sup>1</sup>

- Multipurpose computers
- Specialized embedded devices
- Connected sensors<sup>2</sup>
- Passively trackable objects

Most systems will be composed of multiple types. For example, a connected home system may have:

- A control interface on a smartphone (*multipurpose computer*)
- Heating/ventilation/air conditioning (HVAC) controller, remote control door locks, and blind controllers (*specialized embedded devices*)

---

1 This classification was inspired by Scott Jenson's article "Of Bears, Bats, and Bees: Making Sense of the Internet of Things," *Frog's Design Mind*, August 2012, <http://bit.ly/195mVm4>.

2 In technology terms, these are really just specialized embedded devices. But as we explain on page 39, they are likely to play a different role in the user's experience of the product or service.

- Security sensors such as motion and contact sensors (*connected sensors*)

Table 2-1 summarizes the key differences between these, which are discussed in more detail in the following sections.

**TABLE 2-1.** Summary of connected device types

	MULTIPURPOSE COMPUTER	SPECIALIZED EMBEDDED DATA	CONNECTED SENSOR	PASSIVELY TRACKABLE OBJECT
<b>User interaction</b>	Rich onboard interaction capabilities (e.g., through screens and keyboards)	May have limited inputs/outputs; advanced interactions handled via web/mobile apps	Via web/mobile apps	Via web/mobile apps
<b>Functionality</b>	Generalized; can run wide range of applications	Specialized for specific functions	Single task	Identity only
<b>Processing</b>	Powerful onboard processor	Onboard processor, with some functions provided by cloud service	Mostly in cloud service	In cloud service

## Multipurpose Computers

These are powerful computers designed to perform a variety of computing tasks. They are no longer just PCs, but also smartphones, tablets, and now arguably connected TVs, set-top boxes, and game consoles. They have rich interaction capabilities, via screens, audio, touch input, keyboards, mice, and sometimes voice and gestural interfaces, too. They contain powerful *microprocessors* with external chips for memory and peripheral interfaces, such as networking. We don't think of these as belonging to the category of IoT devices, but they are often used to handle user interactions for IoT services (or sometimes as Internet gateways to devices that don't have a built-in Internet connection).

This book does not cover general design guidelines for PCs, smartphones, tablets, or other multipurpose computing devices. These are huge topics in their own right and many good reference books have been written about them.

## EMBEDDED DEVICES

The falling cost of technology means that a computer is now the cheapest, simplest way to implement even trivial tasks such as timing your tooth brushing. As a result, many more of the objects around us are starting to come with onboard computation and connectivity. This creates some of the most novel and iconic devices of the Internet of Things: the thermostats, bathroom scales, and even plastic rabbits. These are all embedded devices.

Specialized to perform particular tasks, embedded devices come in a wide range of form factors to suit the task and may have other mechanical parts (such as a washing machine) or even other embedded systems as well (such as a car).

Because embedded devices are specialized to do specific jobs, they can do those jobs more reliably and more efficiently than a general-purpose computer system despite having less power. For example, an embedded computer controlling your car brakes can guarantee that it would release and activate the brakes at exactly the right intervals needed to prevent you losing control (this is an example of a real-time system). A general-purpose computer running a general-purpose operating system could perform the same function, but with no guarantees as to how long it would take. The fact that it could also be used to watch YouTube videos is of no use to you when driving on an icy road.

Mobile phones were historically considered embedded devices, as they were highly specialized devices. Modern smartphones are more like general-purpose computers in terms of what they can do for the user, as there are so many ways to extend their functionality through apps.

Embedded systems may need to meet far more stringent operating criteria than general-purpose computers, especially if they are located in inaccessible places or controlling safety-critical systems. They may need to work in harsh environmental conditions (such as down an oil well), conserve electrical power very efficiently to run for years on a tiny battery (as in an environmental sensor), run for years without errors (as in anti-lock car brakes or a nuclear reactor), and potentially recover themselves from failures when human intervention isn't possible (as in an undersea cable). They are also often devices that we expect to set up and largely forget about, such as home boilers/furnaces, which we don't want to have to maintain or reboot regularly. Figure 2-1, Figure 2-2, and Figure 2-3 all show examples of embedded devices.



**FIGURE 2-1**

Nest's thermostat learns how the home's occupants set their heating manually, uses motion and light sensors to detect when the home is occupied, and uses this data to optimize the heating schedule and settings; controls are available on the device or via a smartphone app and web service (image: Nest)



**FIGURE 2-2**

The Withings Smart Body Analyzer bathroom scales transmit weight readings over WiFi to an Internet service for users to track their weight using a smartphone app (image: Withings)

In an IoT context, embedded devices are typically the parts of the system that interact with the physical world, gathering data from it through sensors (like a motion detector) and/or producing physical actions, like a connected door lock or light switch.



**FIGURE 2-3**

The Nabaztag—one of the earliest consumer IoT devices—was a rabbit-shaped ambient device that could read out email and information from Internet services (image: Rama)

Embedded devices may have onboard user interaction capabilities, such as the Nest thermostat, which has a screen, and a bezel that can be rotated to select/scroll and pressed to select. Users can make temperature adjustments and even set schedules on the device itself. But user interactions are often handled at least in part by other devices in the system. The Tado heating controller (see Figure 2-4) has very limited onboard user input/output capabilities: most interactions are handled by a smartphone app. (See Chapter 8 for more information on interface and interaction design for embedded devices.) Note that embedded devices with no input/output capabilities will have no way of providing feedback when they develop a problem, except for ceasing to function. If this is the case, you'll need to ensure the system can use another device (such as a web or smartphone app or SMS) to tell the user there's something wrong and guide them to fix it.

Embedded devices may connect directly to the Internet (as the Nest controller does, over WiFi). They may connect indirectly via a smartphone (as many wearables do; see Figure 2-5) or hub/gateway device (as many home automation systems do; see Figure 2-6). Gateways, and the different ways that devices can be connected, are discussed in more detail in Chapter 3.





**FIGURE 2-4**

The Tado heating controller has limited on-device user interaction capabilities: users control the device through a smartphone app (image: Tado)



**FIGURE 2-5**

The Pebble Smartwatch connects to a smartphone to display app notifications on the wrist; it can also run apps of its own, such as displaying Evernote notes (image: Pebble)



**FIGURE 2-6**

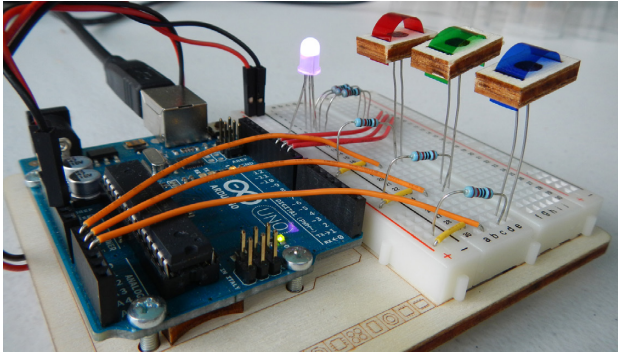
The Hive thermostat can be remotely controlled by mobile and web apps; it connects to the Internet via a dedicated gateway device (the small device in the center; image: British Gas)

### **Embedded Hardware**

An embedded system (the computing part of the embedded device) has some basic types of hardware components in common with a general-purpose computer, such as a central processor, memory, and peripherals (although these components may be integrated onto a single chip called a *microcontroller*, whereas in a multipurpose computer they would often be separate parts).

Microcontrollers typically have far less computing power than multipurpose computers (often just a few KB of memory and a processor clock speed in the low MHz range) but are optimized to perform specific tasks, such as controlling the anti-lock brakes on a car or the functioning of a sewing machine, and to do so with very limited budgets of cost, power, and perhaps space.

In IoT forums, you may often hear the names of some of the common microcontrollers/systems used in prototyping and building IoT devices, such as Arduino<sup>3</sup> (see Figure 2-7), Beaglebone,<sup>4</sup> Electric Imp,<sup>5</sup> Raspberry Pi,<sup>6</sup> and ARM mBed.<sup>7</sup>



**FIGURE 2-7**

Arduino is a popular range of microcontrollers used in IoT prototyping and is designed to be accessible to newcomers to hardware prototyping; this image shows an Arduino Uno used to create a color-mixing lamp (image: Daniel Cortes)

Some of these are designed to be “open” systems: using open source software and hardware. You are given working examples to help you get going, but you can modify these right down to the bare metal if you want. Others (such as Electric Imp) are “closed”—they work out of the box but don’t give you access to their internal workings. They may be easier to use, but you can’t modify them.

### **Embedded Software**

Software for embedded systems is designed to make efficient use of limited resources. A simple device may have only a small, fixed-function program (firmware), just enough to make the system work. This is stored in nonvolatile memory (memory that is not lost when the device

---

3 <http://www.arduino.cc>

4 <http://beagleboard.org/bone>

5 <https://electricimp.com>

6 <https://www.raspberrypi.org>

7 <https://mbed.org>

is not powered) such as Flash or ROM. If you want to change the way the device works, you need to overwrite the firmware (see Figure 2-8). (Flash can be overwritten but ROM cannot, so some devices may have firmware that cannot be upgraded.)



**FIGURE 2-8**

Some Whirlpool laundry appliances featured USB ports to enable users to upgrade the firmware and add new wash programs (image: Whirlpool)

A more complex device may have a specialist embedded operating system and be able to run multiple programs, as desktop operating systems are used to run applications. For example, an OS can manage access to system resources, like the processor or network, if different applications may require them at the same time. An OS also provides flexibility to change the functionality of the device without reinstalling all the software. Real-time operating systems, such as VxWorks, QNX, and RTLinux, are used where time-sensitive processing is needed. Many types of system may require real-time OSes, from spacecraft to robots to WiFi access points (see Figure 2-9). There are even versions of general operating systems specialized for embedded devices, such as Windows Embedded and various flavors of Linux. Operating systems generally require more CPU and electrical power to run than firmware.



**FIGURE 2-9**

NASA's Mars Science Laboratory Curiosity Rover uses the VxWorks real-time operating system, as does Apple's Airport Extreme (images: NASA/JPL-Caltech/Malin Space Science Systems, Apple)

## **CONNECTED SENSORS**

Connected sensors are small embedded devices used for capturing data from the physical world, and passing this to a networked service. Although technically a class of embedded device, they tend to be used and experienced in a different way as part of the Internet of Things. They generally have no onboard user input or output capabilities. So there are no knobs or pushbuttons, and the only way to know what they are doing is via a display on another device, like a smartphone. Sensors tend to be quite low profile, so the devices themselves are often less visible and less salient parts of the experience. The focus of the UX is not on the devices, but on the data that is captured and the service that the data enables (see Figure 2-10).



**FIGURE 2-10**

The NetAtmo Weather Station sensors measure temperature, humidity, air quality, and atmospheric pressure; there are no onboard user interaction capabilities on the sensors, but data can be viewed via smartphone or web apps (image: NetAtmo)

They are often deployed in networks of multiple sensors (in which they may be called “sensor nodes”). A large-scale example such as a network of air quality sensors may have hundreds or thousands of nodes; a smaller-scale example might be a set of home alarm motion/contact sensors. Single sensors may also be practical in some circumstances, such as the Proteus in-body pill sensor (see the Proteus case study in between Chapters 3 and 4), which can be used to detect whether a patient is taking their medication. “Bridging Physical and Digital: Sensors and Actuators” discusses sensor types in more detail.

A connected sensor typically contains just enough onboard computing to gather data and transmit it over a network. This is likely to mean a very basic processing unit with very limited memory and computing power, a communication transceiver, and a battery or other power source (e.g., energy harvesting from the environment, such as solar or wind power). Communications may be a low-powered wireless local area network (WLAN) connection to a gateway, or a high-powered

connection to the Internet via the cellular network. It will generally have no on-device user interface (a home alarm sensor might have an LED to indicate when it is functioning and/or connected to the network). It may be possible to pass simple instructions to the sensor over the network (e.g., to control how frequently readings are sent), or the sensor may only be capable of pushing data to the network and may not be able to receive instructions (see Figure 2-11).



**FIGURE 2-11**

The Proteus smart pill contains a tiny sensor—it has no battery, but is activated by contact with stomach acid, sending a small transmission to a Bluetooth-enabled skin patch, which in turn connects to a mobile phone and notifies relatives or doctors via an Internet service that a pill was taken (image: Proteus Health)

Energy is a particularly precious resource for most sensor nodes and must be conserved as much as possible (see “The Challenge of Powering Devices” later in this chapter for more detail).

Where sensors are deployed in networks, the networks are designed so that the whole network continues to function even if individual nodes fail, or lose connectivity, though data may be lost. A UX design for any IoT system must be able to cope with missing data points. For example, a system that uses sensors to monitor the movement of traffic on a road does not need every data point in order to calculate the speed at which traffic is moving. (See Chapter 13 for more on designing with data.)

## PASSIVELY TRACKABLE OBJECTS

A “thing” can have a simple presence on the Internet without actually having an Internet connection. Passively trackable objects have a unique identity that is associated with information about them online, but are not themselves connected to the Internet.

### RFID and NFC

Simple objects, which need not have any onboard computing at all, can be identified via radio-frequency identification (RFID) or a quick response (QR) code. This unique ID allows the user to access information about the object online.

In terms of data storage, there are two types of RFID:

#### *Read-only RFID tags*

These tags have just enough storage for a unique identifying number, which is associated with more detailed product information in a database, probably on the Internet. If you want to change any of the product information, you’ll have to do so in the database, as you can’t overwrite the data on the tag (see Figure 2-12).

#### *Read/write tags*

These can be overwritten with changing information, so you don’t necessarily require Internet access to retrieve meaningful information from the tag.

RFID tags can operate on different radio frequencies. Higher-frequency tags are capable of transmitting over a longer range, lower-frequency tags transmit over a shorter range. A shorter range is preferable when the tag should be read as the result of an explicit user interaction, such as swiping a smart card or scanning a product label (see Figure 2-13). You don’t want to read the same tag twice, or accidentally read another nearby tag instead. A longer range might be more suited to tracking a number of items over a larger area.

RFID is commonly used in inventory management, to track stock levels in stores and at suppliers. For example, the UK department store Marks and Spencer was an early pioneer of RFID for inventory tracking, using it to keep track of the numbers of each item in each store with the aim of matching supply to customer demand.





**FIGURE 2-12**

A read-only RFID tag of the type used in pet ID chips (image: Wikipedia user Light\_Warrior)



**FIGURE 2-13**

The London Oyster travelcard uses short-range read/write RFID to store local information about the cash or tickets stored on the card

RFID is also commonly used for contactless ticketing and payments, such as debit and credit cards. Disney theme parks issue RFID tickets and wristbands, which visitors swipe to gain access to attractions (see Figure 2-14). The tickets can store data about prebooked attractions and track the movement of visitors around the parks, helping Disney better understand visitor behavior.



**FIGURE 2-14**

The Disney MagicBand wristband, which provides access to attractions and can be used to pay for purchases in the park, helps Disney better understand visitor behavior (image: Ciara Taylor)

Near field communication (NFC) uses the same standard as some shorter-range RFID tags, but evolves it. An NFC-enabled device (e.g., a smartphone) can behave both like an RFID tag (and be read by a reader) or like a reader. RFID uses many different data formats, so different RFID devices cannot necessarily interoperate. A key advantage of NFC is that it provides a single common data format—the NFC Data Exchange Format, or NDEF—which can allow all kinds of NFC-enabled devices and tags to share data.

NFC-enabled smartphones are often used for contactless payments from mobile devices. NFC permits two-way communication between devices, and a potentially interesting application for IoT is in making it easier to set up network connections between other devices. Android Beam uses NFC to streamline the process of setting up a temporary Bluetooth connection between two devices to share data such as a photo or contact (see Figure 2-15). (See Chapter 12 for more details on this.)



**FIGURE 2-15**

Samsung S Beam uses NFC to establish WiFi connections between smartphones (image: Samsung)

QR codes are a very basic way to give a physical device a digital identity. These are two-dimensional barcodes, which can be read by any imaging device able to extract the data encoded in the image, such as a smartphone. Like read-only RFID tags, QR codes rely on the code being mapped to more extensive product information in a database.

An example of QR codes used in a consumer setting is the +More platform created by the UK agency Evrythng for the Diageo drinks company (see Figure 2-16). This assigns each bottle of spirits a unique identifier, which can be accessed by scanning a QR code printed on the label. A Father's Day marketing campaign encouraged customers to buy whisky bottles as gifts, scanning the QR code to record a personalized video message, which the recipient could then in turn access by scanning the code.



**FIGURE 2-16**

The Diageo whisky gifting campaign allowed customers to send a personal video message with each bottle; each bottle is identified with a unique QR code, and scanning the code plays the video (image: EVRYTHING)

## Beacons

Beacons (such as Apple iBeacons) are another type of passively trackable object. They are used to provide very precise location information. Beacons use Bluetooth Low Energy to broadcast a unique ID that can be received by nearby Bluetooth-enabled devices, such as smartphones (Chapter 3 discusses Bluetooth in more detail). The phone looks up the beacon's ID in an online database, which provides information such as who owns the beacon, and exactly where it is. The strength of the radio signal between the beacon and phone helps determine how far the phone is from the beacon. Beacon-enabled apps on the phone can then use that information to provide contextually relevant functions. For example, entering a favorite department store might trigger the loyalty app on your phone to notify you of a special offer or discount voucher (see Figure 2-17). Passing a beacon in the cleaning products aisle of the grocery store might trigger a shopping list app to remind you to pick up dishwasher detergent. Or a beacon in your car might trigger your phone to bring up the option to unlock the car. (More examples of beacon-enabled services are discussed in Chapter 4.)

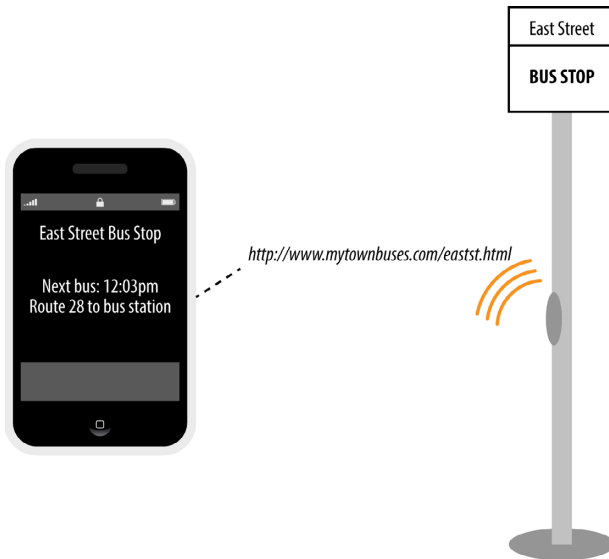
A variant on the beacon concept is Google's Physical Web project. Led by designer Scott Jenson, this experimental project aims to bring interaction on demand to the physical world, without requiring the user to have particular apps installed on their mobile device. Instead of broadcasting a simple ID that must be looked up, or needing to trigger an

app, Physical Web beacons broadcast a web address. Users need not download a dedicated application but can just walk up to any device and interact with it via their phone's browser. As the project's web page cites, bus stops could share information about arrivals, parking meters and vending machines could all be paid for in the same way (see Figure 2-18), and rental cars could broadcast a sign-up page, allowing users to sign up and drive away quickly.



**FIGURE 2-17**

An Estimote beacon triggering a smartphone app to display a special offer (image: Estimote)



**FIGURE 2-18**

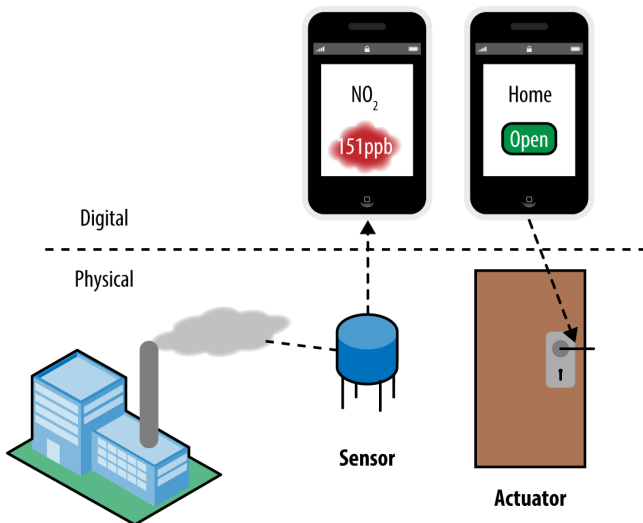
A Physical Web device broadcasts the web address through which the user can access its information or controls

## Bridging Physical and Digital: Sensors and Actuators

Sensors and actuators are the device components that bridge the Internet and the physical world. Sensors convert energy readings from the physical environment into numeric values that can be transmitted digitally. Actuators convert digital instructions into mechanical actions. (See Figure 2-19.)

### SENSORS

Sensors receive inputs from the physical world (e.g., motion, light, air quality, contact, location, proximity, humidity, orientation, etc.). They detect the presence of energy or changes in energy and quantify it, producing a numeric value. A digital thermometer usually converts heat energy to a voltage, and then quantifies the voltage as a temperature reading. Similarly, a photosensor might convert light energy to a voltage then output a numeric reading.



**FIGURE 2-19**

Sensors convert readings from the physical environment into digital information; actuators convert digital instructions into mechanical actions

Sensors can be active or passive. Active sensors inject energy into the environment to detect changes of some sort (see Figure 2-20); passive sensors detect energy that is already there. The motion detection systems used in stores (or to protect priceless museum exhibits in heist

movies) use beams of light shone across a room at photosensors: when something (such as a robber) breaks the beam, the sensor measures the loss of light.



**FIGURE 2-20**

The Samsung Galaxy S5 has an active sensor that can be used to measure heart rate; it fires a red light through the user's finger, using changes in light level to measure changes in blood flow that indicate a pulse (image: Samsung)

Motion detectors used in home security alarms (such as Figure 2-21) tend to be passive infrared sensors: human beings radiate infrared energy with a wavelength of around 9–10 micrometers, so alarm sensors are tuned to respond to rapid changes on this wavelength that indicate a person moving around. (Slower changes might simply mean that the room is warming up as the sun moves around.)

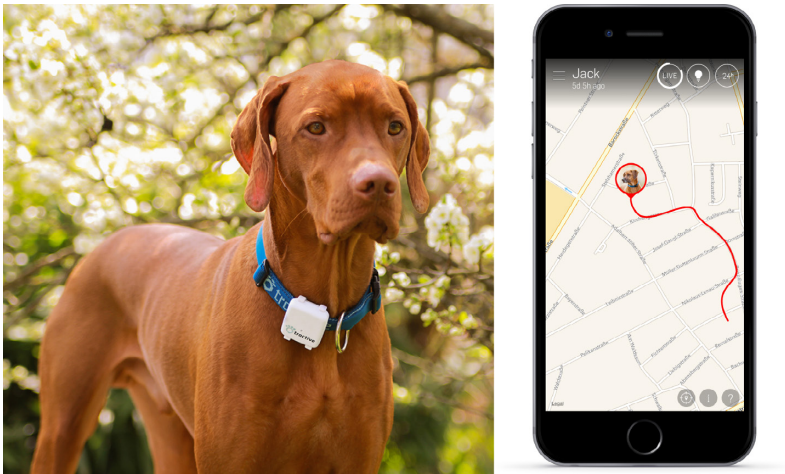


**FIGURE 2-21**

The Dropcam connected camera contains a passive infrared sensor to detect motion (image: Dropcam)

Some sensors measure the presence of chemicals. An air quality sensor might measure harmful volatile organic compounds (VOCs), such as benzene from car exhaust, by detecting changes in conductivity caused by the chemicals reacting with the surface of the sensor.

Location sensing is commonly used in applications for mobile devices. A GPS receiver calculates its position by timing transmissions received from GPS satellites (see Figure 2-22). The tiny compasses found in mobile phones use magnetic field sensing to tell which direction is north. Bluetooth beacons, described earlier, can be used to enable indoor location detection based on the proximity and signal strength of the beacons.



**FIGURE 2-22**

Tractive pet trackers track the location of your cat or dog using GPS (images: Tractive)

Accelerometers measure acceleration based on vibration (see Figure 2-23). They are often used in mobile phones and tablets to detect screen orientation and present the UI in landscape or portrait mode. Some alarm clocks use accelerometers to identify the best time to wake the user. The clock is placed on the mattress and detects differences in the sleeper's movement during different sleep phases, waking them during light sleep. Gyroscopes measure angular acceleration (rotation around an axis), and can be used for more accurate movement measurement than an accelerometer alone.





**FIGURE 2-23**

The Wii Remote uses an accelerometer to detect rotation; the WiiMotionPlus (the attachment on the bottom) adds a gyroscope for more accurate movement detection—this is used in many sports games and for more “realistic” sword fighting in “The Legend of Zelda: Skyward Sword” (images: Nintendo and Evan Amos)

Sensors are increasingly found not just in networked embedded devices, but also in mobile devices. There are many applications (such as activity tracking) that formerly required a separate sensor but can now be achieved using a smartphone’s onboard sensors (see Figure 2-24). However, a dedicated sensor (such as a wearable tracker) may be more accurate or a better fit for the context of use (e.g., a phone carried in a handbag may move differently from a phone carried in a pocket as the user walks).



**FIGURE 2-24**

The Withings iPhone app can now track the user’s steps using the iPhone’s onboard accelerometer; however, Withings still makes dedicated activity trackers, such as the Activité (image: Withings)

## ACTUATORS

Actuators provide the means for a digital system to act on the environment. They convert electrical energy into mechanical energy, producing movement in the real world. This might control a motor, or simply turn something on or off. Connected door locks (see Figure 2-25), ceiling fans, and window shade motors (see Figure 2-26) are all examples of consumer actuators. Even bubble machines have been used as actuators (see Figure 2-27).



**FIGURE 2-25**

The Danalock connected door lock allows users to lock and unlock a door from a smartphone (image: Danalock)



**FIGURE 2-26**

The Fortrez water shutoff valve can be programmed and controlled remotely via a home automation app (image: Fortrez)



**FIGURE 2-27**

The Bubblino Internet-connected bubble machine is a fun example of actuation: it watches Twitter and turns on the machine to blow bubbles when it sees a particular hashtag (which the user can define; image: Adrian McEwen)

## The Challenge of Powering Devices

One of the key considerations in the technical design of a connected device is how it gets, and uses, power. This can have a fundamental impact on the functionality of the system and the user experience.

## Conserving Battery Life

Many embedded devices and connected sensors run on batteries. This may be because:

- They need to be portable
- They need to work where mains power is not available (e.g., outdoors or underwater)
- It makes them easier to install (avoiding the need to rip holes in your walls or call out an electrician)
- It's not safe to run mains power to their location (e.g., the device that takes readings from smart gas meters runs on a battery, as it's not safe to run a power line near a gas pipe)

Many sensors are small, so will have only tiny batteries, capable of holding limited energy. As mentioned earlier, some devices may run on energy harvesting from the environment, such as solar or wind power, but this can be unreliable (it's not always sunny or windy). A few may be able to generate enough power from a user interaction—for example, pressing a switch could generate a tiny amount of power from a piezo crystal that might be enough for a tiny data transmission (see Figure 2-28).



**FIGURE 2-28**

The Philips Hue Tap light switch is powered by kinetic energy generated when the switch is pressed (image: Philips)

Many embedded devices and sensors are background devices: things we want to set up and think about as little as possible. Some of them may be in hard-to-reach places, and if we have any number of them, we don't want to be thinking about charging or battery changing any more often than we have to. If you have a battery-powered smoke alarm, you don't want to be changing the battery more than once every year or two. In some cases, it may be extremely difficult to reach the device to change the battery, such as an internal pacemaker monitoring the vital signs of a patient with a chronic heart condition.

Processing data and connecting to networks drains energy, so devices that need to conserve energy need to do as little processing and as little talking to the network as possible. Embedded software is often programmed such that the device spends as much time as possible in a

sleep or low power mode, waking up the processor only occasionally when it needs to do something (see Figure 2-29). For example, a radiation sensor might take readings in low power mode and wake up to send them at regular intervals, or it might wake up and connect to the network more frequently when the sensor reads over a certain threshold. Maintaining a constant connection uses a lot of power, so is best reserved for mains-powered devices.

This need for energy conservation is the main reason why many edge devices—embedded devices and sensors—are not directly connected to the Internet. Internet connectivity, such as cellular data and WiFi, requires a lot of power. Local, lower-bandwidth networks such as Bluetooth LE and ZigBee use much less. There are also a number of communication patterns and protocols that can be used to manage network connections, and (to an extent) help to control power consumption. These are described in the “Types of Network” and “Network Communication Patterns” sections in the following chapter.



**FIGURE 2-29**

The Lowes Iris home security motion sensor transmits its status at two-minute intervals as long as there is no motion; if it detects motion, it will wake up and transmit this information right away (image: Lowes)

## **POWERING DEVICES CREATES UX CHALLENGES**

UX designers don't normally have to concern themselves with the electricity consumption of devices. But, in IoT, the secondary effects of trying to conserve energy consumption ripple all the way up to the UI level.

Devices often connect only intermittently. That means that communications are essentially asynchronous: it may take time for a message to reach all the devices in a system. Some data may be out of date or missing, and in some cases, different devices will give you different information about the status of the system. Whether and how much this matters depends on the type of service you're building. (For more information on networking and continuity, see Chapter 3 and "Continuity" in Chapter 9.)

Power consumption is often at the root of some of the responsiveness challenges that IoT services pose for UX designers: intermittent connectivity and asynchronous communications (see "Networking Issues That Cause UX Challenges for IoT" in Chapter 3 for more details).

## Summary

Edge devices are the *things* in the Internet of Things.

Many IoT devices are *embedded devices*: custom hardware and software specialized to perform a specific function. Connected sensors are a type of small embedded device, often deployed as part of sensor networks.

Sensors and actuators are the bridges between the Internet and the physical world. Sensors convert energy readings from the physical environment into numeric values that can be transmitted digitally. Actuators convert digital instructions into mechanical actions.

Embedded devices and sensors often have limited computational power and many run on batteries, so must make efficient use of processing and electricity. Maintaining a network connection requires a lot of energy, so many IoT devices only connect to the network intermittently. This can cause discontinuities in the user experience, where instructions take time to reach edge devices or some parts of the system may have more up-to-date information than others.

# Networks: The Technology of Connectivity

**BY CLAIRE ROWLAND**

This chapter explores networking: how the things communicate with one another. Designers take this for granted when working with multipurpose computers. But networking for connected devices is different, in ways that can have a radical impact on UX. There are different patterns of network connectivity in IoT; each has advantages and disadvantages.

If you're a designer without a technical background, a basic working knowledge of how data flows around IoT systems will help you anticipate some of the challenges you may come across.

If you're an engineer, you will already know much of what's in this chapter, but you might not have thought about how it can affect the UX.

This chapter is lengthy, but it's designed as a reference. If you're keen to get on to the design chapters, just read the first section for now to understand why networking matters to UX. Come back to the rest later, when you want to know more about the underlying technology.

This chapter introduces:

- The impact of networks on the UX of an IoT system (see page 58)
- Some typical architectures of IoT systems (see page 66)
- The basics of different types of network used in IoT, both Internet and non-Internet local networks (see page 75)
- Network communication patterns, which govern how data flows between devices (see page 87)

- The role of the Internet service and application programming interfaces (APIs) (see page 93)

It will help you understand:

- How network latency and reliability can impact UX (see page 60)
- How choosing local or Internet connectivity shapes system functionality and UX (see page 71)
- How gateways are a burden to develop but also often a necessity (see page 72)
- Why many edge devices cannot connect directly to the Internet, but how this may change (see page 85)
- How push, pull, and polling communication patterns determine whether a device has up-to-date system data (see page 87)
- How API design and UX design are related (see page 96)

## Why Is Networking Relevant to IoT UX?

When we design for PCs and smartphones, we tend to assume that under normal conditions:

- A good network connection will be available (e.g., broadband or fast mobile data)
- The device will be constantly connected to the network
- The system will feel responsive to the user's actions
- Where the system can't fulfill the user's request immediately, it can provide good feedback on progress
- If we're accessing a service via more than one device, all those devices will always be in sync and be up to date with any service information

Both designers and users know that connectivity doesn't always work perfectly. Mobile coverage can be patchy in places, emails or web pages can sometimes be slow to download, and Skype calls can fail. But things mostly work well enough that we design for a constant, working connection and quick response times as the default state. And when things don't work, designers can usually mitigate the damage to the UX by:



- Providing progress information (e.g., for slow downloads)
- Saving data to let users resume interrupted actions (e.g., an e-commerce checkout process)
- Providing information on what has gone wrong, how much of the intended action was successful, and what the user might be able to do about it (e.g., sending an email delivery failure message)

Designing for IoT is different. Connected devices may use different types of network and have different patterns of connectivity. We cannot assume that they will be constantly connected, or instantly responsive. States that might be edge cases in conventional software UX (device offline, data that is not right up to date) may become core considerations for your IoT UX. On top of this, users may be unprepared for the way that Internet networking will affect their interactions with familiar everyday objects.

The following sections describe the impact that networking can have on UX. If you're a designer accustomed to working with multipurpose computers, you may not have encountered some of these issues before. A basic understanding of networking will help you work with engineers to anticipate potential glitches, and the best way to handle them in your design. If you're an engineer familiar with IoT technology, you'll already be familiar with the technical content of this chapter. But you might not have considered how networking can affect the user's experience.

## Networking Issues That Cause UX Challenges for IoT

The key networking issues that affect UX in IoT are: intermittency, latency and responsiveness, and reliability.

### **IOT DEVICES OFTEN CONNECT ONLY INTERMITTENTLY**

Many IoT devices only connect intermittently, often as a way to conserve power (as discussed in Chapter 2). This can result in parts of the service being out of sync with other parts. The UX can then feel like a bunch of devices doing different things instead of a single coherent whole (see Chapter 9 for more information).

## LATENCY AND RESPONSIVENESS MAY VARY

The Internet gets data from one point to another in a way that chooses reliability over speed. When you send (or request) a piece of data, you know the start and end point of its journey. However, you have no control over the route it will take to get there. This is intentional: if parts of the network go down, your message can be routed via other parts and will probably get through. However, it also means that there are no guarantees as to how fast it will get there. The result is that *latency* (the time it takes for your message to pass through the network) is out of your control. You could be sending an email to someone on the next street, but the Internet might route your commands halfway around the world.

This inability to control latency is a big challenge for UX because it affects *responsiveness*. Instant responses cannot be guaranteed when interactions are routed via the Internet.

We expect this delay from conventional Internet interactions. It's frustrating if your email takes a minute to download, but you know this happens sometimes. At least you usually have some feedback that it is in progress.

By contrast, we expect physical things to respond to us immediately. When you turn on a physical light switch, the bulbs illuminate almost instantly. If you tried to turn on your living room lights from your phone and nothing happened for 30 seconds, that would be a poor experience. But there is no way to guarantee immediate responsiveness over the Internet (see Figure 3-1).

Jakob Nielsen's 1993 book *Usability Engineering* sets out guidelines for the responsiveness of a system.<sup>1</sup> A UI should respond to a user command within 0.1 seconds for no delay to be noticed. A delay of between 0.2–1.0 seconds will be noticed, but users think the computer is working on the problem. Delays of more than 1 second require the computer to display feedback to show that it is working on the problem. Nielsen's guidelines were written in an age when a system was often a single device that may not have been networked. But human perception doesn't change. The light might not turn on immediately because

---

1 Jakob Nielsen, *Usability Engineering*. Morgan Kaufmann, 1993.

it takes time to send the command across the network. But the control UI must acknowledge the user's action right away. (See the smart plug example we look at in “The Impact of Latency and Reliability on UX.”)



**FIGURE 3-1**

The design and product company BERG produced a beautiful concept prototype for a connected washing machine.<sup>2</sup> The video shows instant responses between the mobile app and washing machine, running over the Internet. In a real world situation, this could not be guaranteed (images: Timo Arnall of BERG).

Even 1 second to turn on a light could feel noticeably slow to someone not expecting it. And if the user has time to be surprised by the delay, the interaction has claimed their attention in a way it shouldn't have. But the nature of the Internet means that it will not be possible to guarantee a response time. It may also not be possible to provide insightful feedback that your action is in progress. (See “Networks Are Not 100% Reliable.”)

In our view, this will make users far less tolerant of poor responsiveness from consumer embedded devices than sluggish mobile and PC interactions. Engineers will need to find ways of minimizing latency for important interactions. Designers will need to find ways of managing consumer expectations and designing around the limitations.

---

2 BERG, 2013, Cloudwash <http://vimeo.com/87522764>.

## NETWORKS ARE NOT 100% RELIABLE

Another aspect of network communication that may feel strange in the physical world is *reliability*: whether or not a message gets through. If you flip a light switch, you expect something to happen: the light does not sometimes “lose” your instruction. But a command sent across a network can fail to arrive.

The Internet is designed to maximize reliability and provide feedback on whether or not a message arrived at its destination. (See “Types of Network.”) But connected devices may use local networks that don’t have the same built-in safeguards. It’s impossible to *guarantee* that a user command won’t get lost. And it’s sometimes impossible to provide confirmation that the command has been executed.

If commands can go missing, and the user doesn’t necessarily know whether they got through or not, the value of the service is undermined.

All networking is unreliable to some extent. This means that edge devices and gateways (where used) may need to *buffer* (temporarily store) data when the network goes down and resync once back online.

## THE IMPACT OF LATENCY AND RELIABILITY ON UX

The impact of latency and networking reliability on the UX depends on the type of service you are building. For a large network of pollution sensors, the sensors don’t need to be connected all the time. It’s OK if data does not get through quickly, or even if data points occasionally go missing. But when you are sending instructions to a device such as a light, you expect it to respond quickly.

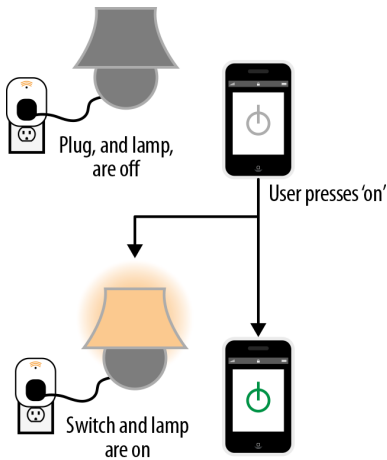
Latency and reliability can also affect design decisions at the UI level.

Take the example of a smart plug controlled by a smartphone app. Pressing a button on the app toggles the plug on and off. This feels like an easy and natural metaphor.

This requires two UI widgets in the smartphone app:

- One to indicate state (is the plug switched on or off)
- One to change state (turn the plug on or off)

The obvious design is to mimic a physical switch and have them be one and the same widget. If it shows “on,” then the switch is on, and touching it will change it to “off” and turn the switch off (see Figure 3-2). This is simple and intuitive, given our everyday experience with switches.



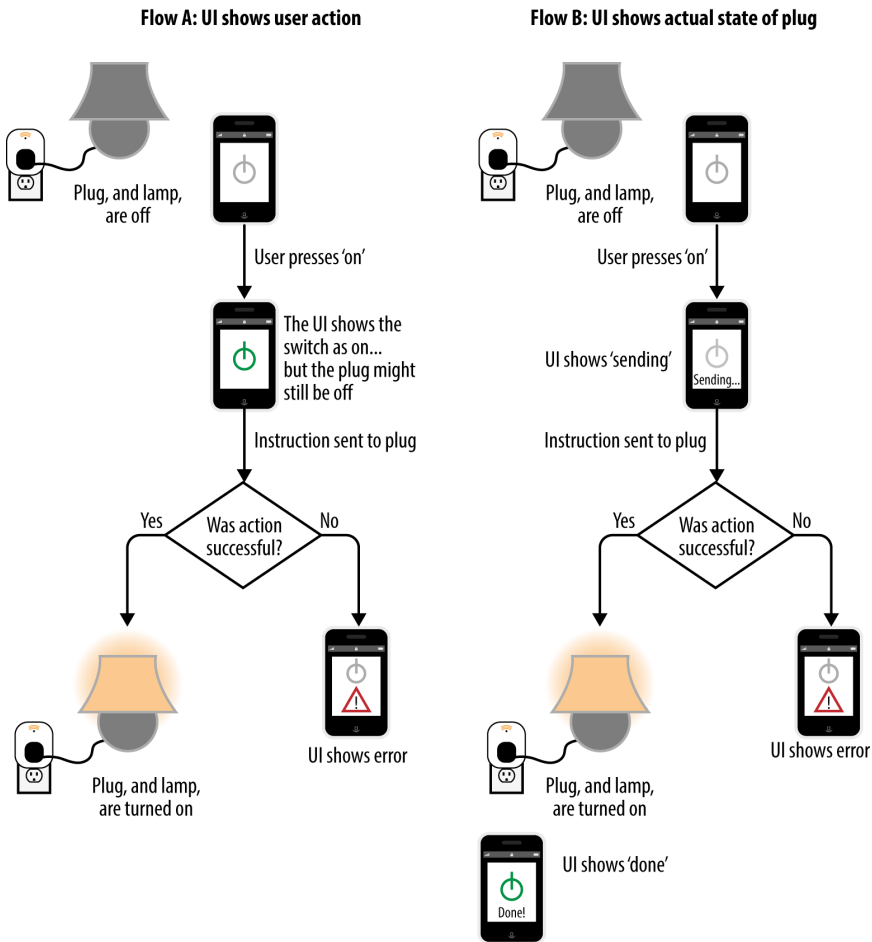
**FIGURE 3-2**

How we expect switches to work

However, network unreliability and latency can raise questions for the UI design. Does the widget indicate what you've *asked* to happen, or what has *actually* happened?

The obvious design will be responsive, but may not reflect reality. The widget will show that the plug is on *before* the physical plug turns on (this may be only a few seconds, but it could be longer). If the switch is currently offline, the app will indicate that it is off. In fact, it might still be on, just not connected to the network and therefore unable to receive the instruction.

An alternative design might separate these widgets: so one indicates the actual state of the plug and one changes it. If the plug is offline, the UI can show that the current state is not known. This tells us the accurate state of the switch, but means we have to represent in the UI the intermediate states of “off-but-asked-to-turn-on” and “on-but-asked-to-turn-off” (see Figure 3-3 for a comparison of the two approaches, and Figure 3-4 for an example of a UI that uses the second approach). This is a lot more complex than a simple on/off switch! (For a more detailed discussion on handling this issue in UX design, see “Continuity” in Chapter 9.)



**FIGURE 3-3**

Alternative UI approaches for displaying the state of the device

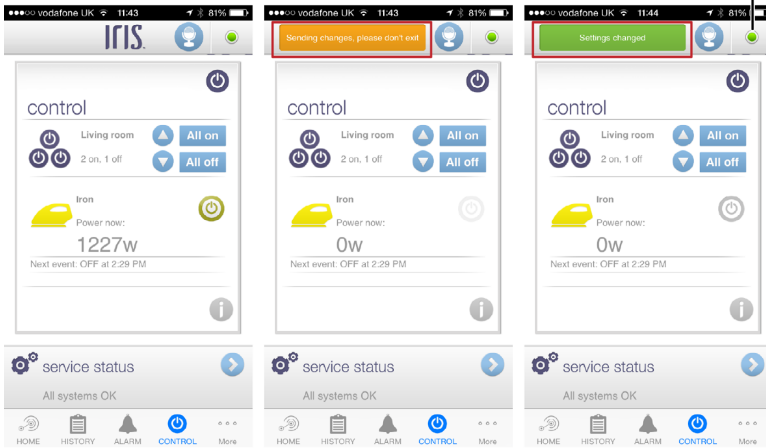
## OTHER WAYS NETWORKING CAN IMPACT UX

The choice of networking technology that your system uses can affect the UX in other ways:

### *Ease of installation*

How easy is it to install a new device? This affects the overall usability of the system. The process of connecting devices to the system (pairing) is slightly different for different network technologies. See Chapter 12 for more information.

This icon provides yet more information on the status of the system. It indicates that the app is able to access the Internet service and the hub (gateway).



1. A smartplug attached to an iron is turned on.
2. The user turns it off from the smartphone app. The power switch icon pulses and a message is shown indicating that the instruction is being sent.
3. The system then confirms that they are applied.

### FIGURE 3-4

The Lowes Iris UI uses an onscreen message to show when a command is being sent; it updates the state of the plug when it receives confirmation that the action was executed (images: Lowes)

### Interoperability

Does the device work with the other devices you want it to work with, and how well does it work? Different types of device may support different types of network, and not all will work together well, if at all. See Chapter 15 for more information.

### Addressability on the Internet

Is the device you want directly addressable on the Internet, or does it need to be connected via a gateway? A fully Internet-enabled device may be able to interoperate better with other devices, but may require you to think more about security.

## [ NOTE ]

The rest of this chapter goes into more technical depth around the networking issues that can affect the UX of an IoT system. Of necessity, this is quite long and detailed. If you're keen to start reading about design (or you already know about IoT networking) feel free to skip ahead to Chapter 4, and return later as needed.

## The Architecture of the Internet of Things

The system architecture defines the path through which one device connects to another. This path determines how flexible the system can be, and how responsive and reliable it is likely to be.

This section gives enough background on system architectures to underpin the issues discussed here and in the chapters coming up. It's worth understanding some common types of network you might encounter, as each has different implications for UX.

As a general overview, some devices may be directly connected to the Internet. But many are not, and instead connect locally to a *gateway*, which then connects out to the Internet.

A *gateway* is a combination of the hardware and software needed to link two different types of network (Figure 3-5). IoT gateways typically translate between Internet Protocol (IP)-based communications and local, non-IP networks used to connect the gateway to the edge devices. The gateway is the furthest point to which the proper Internet reaches. (It's different from your broadband/WiFi router, which connects devices that support IP networking directly to the Internet.) Consumer IoT gateways are often called *hubs* or *bridges*.

We discuss the types of network in more detail in “Types of Network.” First, this section describes some common architectures for IoT systems, and some of the advantages and disadvantages of each for UX and product design.

### DEDICATED GATEWAY

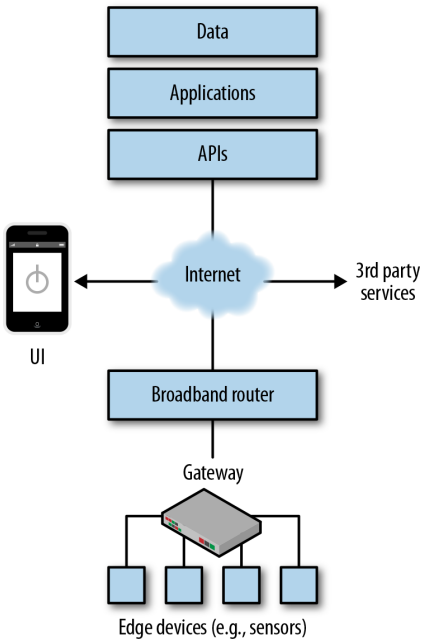
Figure 3-6 shows a system architecture with a gateway device. This is how a home automation system (like SmartThings) works.





**FIGURE 3-5**

The SmartThings hub and Philips Hue bridge are both examples of gateways (images: Philips and SmartThings)



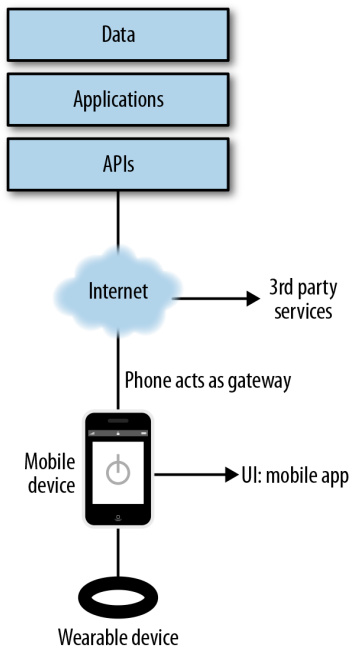
**FIGURE 3-6**

A typical architecture for a home automation system—edge devices (such as sensors) connect to a gateway (or hub) device, which connects in turn to the Internet service over broadband (there may be a SIM card for cellular data backup if the broadband goes down); a sensor network might follow a similar architecture

*Edge devices* (the end points of the system) such as light switches and security sensors use a low-powered network protocol such as ZigBee or Bluetooth to talk to the gateway. None of the edge devices are directly accessible on the Internet: all inbound and outbound communications must pass through the gateway. The gateway translates between the ZigBee protocol and Internet Protocols (we'll discuss this in more detail momentarily) and provides a security firewall, which controls incoming and outgoing communications to protect the edge devices from malicious attacks. It can also act as a local brain for the system—for example, running the security system or home automation rules.

### SMARTPHONE AS GATEWAY

A smartphone can also act as a gateway (see Figure 3-7). This is a common pattern for wearable devices, such as the Pebble Smartwatch. The diagram shows a wearable connecting to a smartphone (most likely via Bluetooth). The smartphone has an onboard app that interacts directly with the device. It also acts as a gateway to translate between Bluetooth and Internet networking, via cellular data.



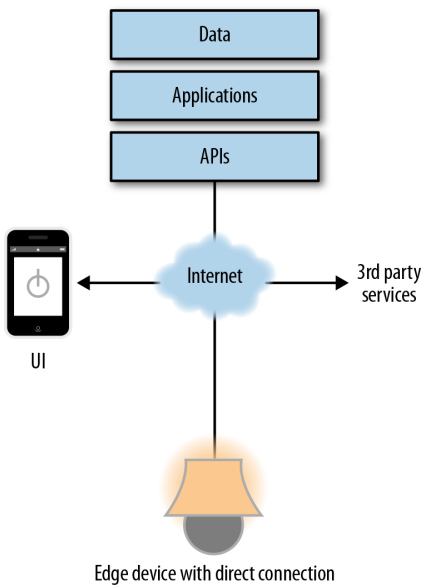
**FIGURE 3-7**

Many wearables use a smartphone as a gateway

Using a smartphone as the gateway means you don't need to provide a separate gateway device (as long as your target audience owns a smartphone). However, a smartphone is not a suitable gateway for any device that needs connectivity when the user is not around. A connected home system that needs to function when the house is unoccupied (e.g., to run a security alarm) needs a gateway that is permanently in the home.

### DIRECT INTERNET CONNECTION

In some cases, an embedded device might connect directly to the Internet service via WiFi or cellular data without using a gateway (see Figure 3-8). For example, all Belkin WeMo devices connect in this way. These devices nearly always have mains electrical power (WiFi uses a lot of power).



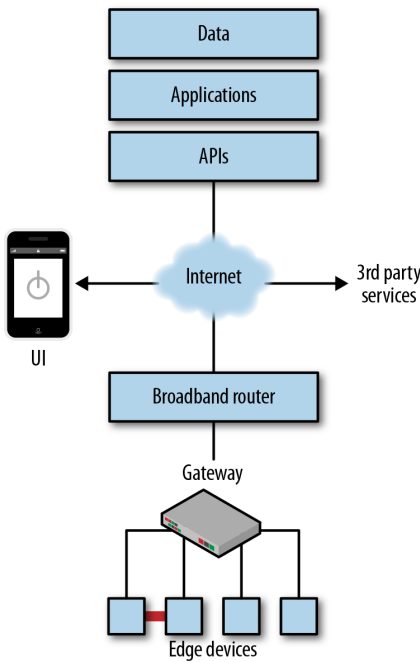
**FIGURE 3-8**

Some devices connect directly to the Internet using cellular data or WiFi (WiFi router not shown)

### DEVICE-TO-DEVICE CONNECTIONS

Edge devices can sometimes connect directly to each other without going via the Internet or a gateway (see Figure 3-9). So, for example, an intercom system might connect directly to a nearby speaker to sound the alert that someone is at the door. It might also send the alert to the

Internet service via a gateway so that if you're not at home, you can receive alerts and perhaps talk to the visitor. But the connection to the speaker goes direct rather than via the gateway.

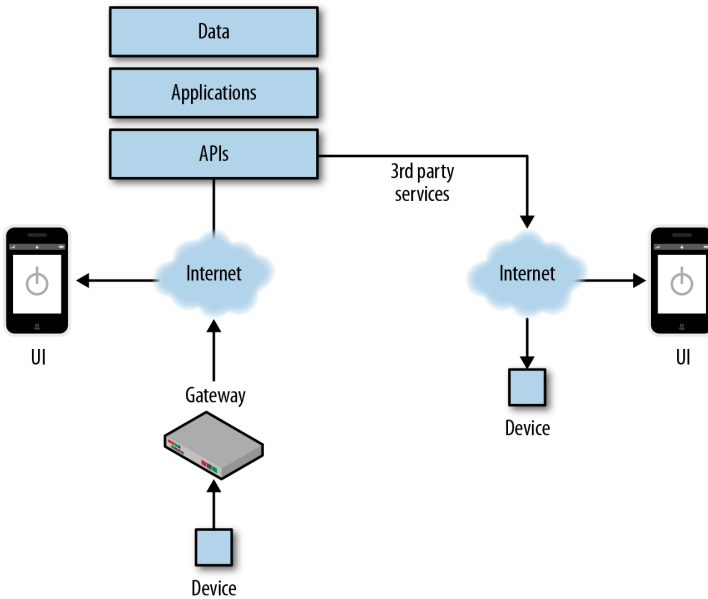


**FIGURE 3-9**

Edge devices can sometimes connect directly to each other (home example with broadband router shown)

## SERVICE-TO-SERVICE CONNECTIONS

Internet services can connect together over application programming interfaces (APIs). (APIs are discussed later in “Internet Service.”) Devices from different manufacturer services can be integrated without the need for a shared gateway (see Figure 3-10). This is how Nest works with API partners. For example, your Jawbone activity monitor could spot when you're waking up and notify the Jawbone Internet service. This then notifies the Nest Internet service, which sends a message to your thermostat to turn on and warm up the house.



**FIGURE 3-10**  
Internet services can connect together via APIs

## HOW SYSTEM ARCHITECTURE AFFECTS UX

As explained earlier, the choice of system architecture determines the flexibility, responsiveness, and reliability of the system.

If a sensor is directly connected to an *actuator* (e.g., a motion sensor connected to a camera), the behavior should be quick and robust. But it is also inflexible: there's not much scope to add or change system functionality.

If the path from the sensor to the actuator goes through a gateway, there are double the number of wireless connections to break and more code that could contain bugs or cause latency. But there is more flexibility. The sensor can trigger other devices connected to the gateway. The gateway software enables smarter coordination across multiple devices.

If the path goes through the Internet, there is greater potential for unreliability and latency, but the system is more flexible yet. For example, the sensor could now be used to trigger an action on any other Internet-connected device, and the camera could stream video to a smartphone anywhere in the world.

It is possible to do all of these in parallel, to get the best of all worlds: fast, reliable local connections with the flexibility and remote access of the Internet. But this needs careful system design to ensure that everything stays in sync.

Right now, many edge devices can't run the high-powered networks required to give them a direct Internet presence. But this is likely to change in the future. There is a lot of interest in the technical community in enabling Internet communications over low-powered networks. We'll look at this in more details in "IP to the Edge."

## **ADVANTAGES OF GATEWAYS**

Gateways (aka hubs or bridges) are an essential part of many IoT systems. They enable devices to work together and can help simplify the UX.

### **Edge devices can be kept simple**

Internet networking and security (such as encryption and decryption) require significant electrical and processing power. Handling these in the gateway allows edge devices to be kept simpler and cheaper. They can run on more basic electronic components and you don't have to write as much code for them. And if you want to control who can access the system, it's much simpler to centralize user authentication in the gateway. No one wants to have to log into a light switch.

Security is discussed in more detail in Chapter 11.

### **Easier setup and maintenance**

Gateways simplify installing and maintaining multidevice systems.

If you have a lot of devices that need to connect individually to the Internet, you will have to set up the Internet connection for each one. Imagine all your light switches were WiFi connected. If you changed the password on your WiFi, you would have to reconfigure each one individually. If you moved house, the new occupant would have to set up the light switches on *their* WiFi network one by one.

With a gateway device, each light switch could be paired with the gateway at the touch of a button over a local radio protocol such as ZigBee or ZWave. You would leave the gateway behind when you moved. The new occupant would have to connect the gateway to their new WiFi network, but they would not have to update every single device.

### **The system can function when the Internet is unavailable**

A gateway can provide a local source of control and decision making that can keep the system running even when the Internet is unavailable. For example, a home automation gateway often stores “rules” locally, so your lighting schedule won’t stop working if the Internet goes down.

### **Lower latency**

You can’t control how data is routed over the Internet, but you can over a local network, meaning latency will be much lower. When you need the system to be responsive, such as turning off the electricity when a gas leak is detected, local connectivity is less risky than routing commands over the Internet.

### **Enabling interoperability**

Gateways can be used to bridge different edge devices that might use different protocols or types of connectivity. The devices don’t need to connect to each other: as long as the gateway can work with each of them individually, they can be used together. So you could combine your ZigBee thermostat, WiFi electrical sockets, and ZWave shade controllers on the same system and access them via the same mobile app. The gateway handles the coordination between the different devices.<sup>3</sup>

It should soon be possible to run Internet Protocol (IP) networking over low-powered networks. Bringing the Internet to edge devices might simplify interoperability even further. Instead of having to translate between different network types, all devices would be able to speak a common language. But this isn’t yet an option (see “IP to the Edge”).

## **DISADVANTAGES OF GATEWAYS**

However, gateways also have disadvantages. The lack of common technical standards means they must be custom built for each system, and centralizing all communications through a single point carries risks.

---

3 This doesn’t necessarily mean that it will be easy to get the devices to work together in all the ways that you want, as the different standards may support different types of behavior in different ways. But it does mean you can connect them to the same system and talk to them via a single UI. For more detail on designing for interoperability, see Chapter 10.

### **Time consuming and costly to develop**

There are no standard ways to translate between IP networking and local networks. So each service provider creates their own proprietary gateway to meet their own needs. Engineers can't reuse code written by other engineers, which raises the cost of development. For users, the lack of a standard approach means that different systems won't work together. And multiple gateways can quickly use up all the Ethernet ports on their broadband router.

### **A potentially confusing point of failure**

If all communications have to go through the gateway, then the gateway is a potential point of failure. If it stops working, so does the rest of the system.

The failure might be technical, or it might be user created. Consumers often don't understand what the purpose of the gateway is. Unwitting household members may unplug the "mystery box," as they don't realize what it's doing (see Figure 3-11).



**FIGURE 3-11**

The AlertMe smartplug is an electrical socket that monitors energy usage and allows remote on/off control. It also has a ZigBee chip for extending the range of the home network via mesh networking. Devices like this are at risk of being unplugged and moved by others in the home who don't realize they are integral to the network. Other devices may then lose connectivity.



As connected devices become more mainstream, gateways may be incorporated into more familiar devices such as home routers or set-top boxes. Table 3-1 summarizes the advantages and disadvantages of gateways.

**TABLE 3-1.** Advantages and disadvantages of gateways

ADVANTAGES OF GATEWAYS	DISADVANTAGES OF GATEWAYS
Handling Internet networking and security in the gateway allows edge devices to be kept simpler and cheaper.	Gateways are time consuming and costly to develop, due to the lack of standards.
Gateways make it much easier to install and maintain multidevice systems: the user doesn't have to set up the Internet connection for each device individually.	Gateways add another potential point of failure to the system: either technical or user created (e.g., if the gateway is unplugged).
A gateway can provide a local source of control and decision making that can keep the system running even when the Internet is unavailable.	
Lower latency: commands can usually be passed more quickly between two local devices over a local network than over the Internet.	
Gateways enable devices that support different network protocols to interoperate.	

## Types of Network

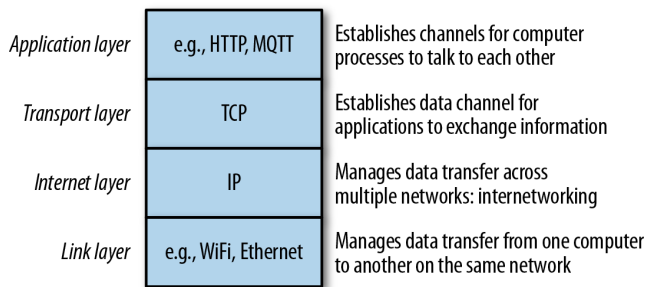
If you're not already familiar with how networking works, this section will help you better understand how technology can shape UX for IoT and how this may change in the future.

In this section, we'll look at:

- How the Internet works
- The pros and cons of different Internet and local network technologies
- The potential for bringing direct Internet connectivity to a wider range of devices

## HOW INTERNET NETWORKING WORKS

Internet networking is composed of different layers, shown in Figure 3-12.<sup>4</sup> At each layer, there are different protocols: agreed systems of digital rules for exchanging data between computers. Each layer is independent of the lower layers. This means that the application (and the user) doesn't need to know which low-level communications protocols are being used. For example, when downloading email on your smartphone, you don't need to know whether it's using cellular or WiFi, and neither does the email application. This separation of concerns simplifies design and engineering.



**FIGURE 3-12**

The Internet networking stack

*Link layer* protocols manage data transfer from one computer to another across a single network.

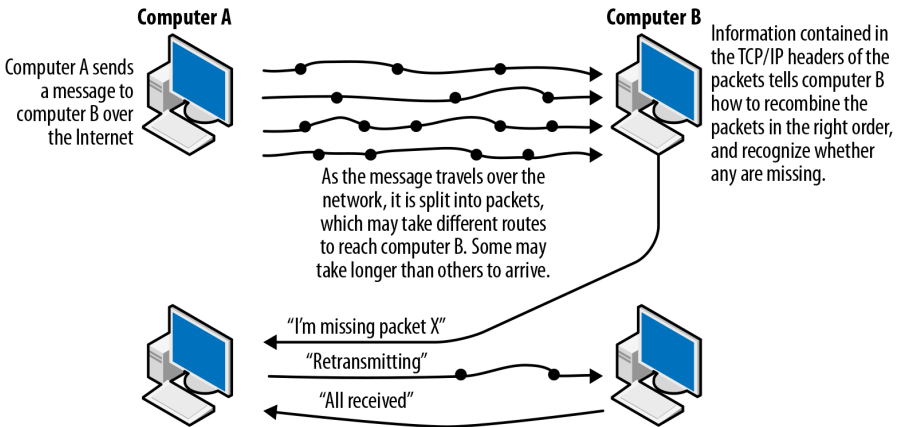
The *Internet layer* uses *Internet Protocol (IP)* to manage the transfer of data across multiple networks, called internetworking. At this layer, devices are given a unique identity on the Internet, the IP address. All Internet communications run over IP. If a device supports IP networking, it can connect to the Internet.

The *transport layer* establishes a data channel for applications to exchange information with each other. The most common transport level protocol is Transmission Control Protocol (TCP). This allows messages to be split over multiple packets and recombined on receipt, even if they have arrived via different routes. It also provides ways for the

---

<sup>4</sup> This is the TCP/IP model. The more comprehensive OSI model sets out seven layers. But for our purposes, the simpler TCP/IP model is enough to understand how the Internet works.

sender to know which packets arrived and which went missing, so the missing ones can be retransmitted. TCP is a reliable way of transmitting messages and ensuring they get through without errors, but this comes at the cost of potential delays. TCP over IP (TCP/IP) is the basis of most of the Internet (see Figure 3-13). If a device supports TCP/IP, it's possible to track whether it received your message or not.



**FIGURE 3-13**

TCP/IP prioritizes the reliability of data transfer over speed

The topmost layer of Internet networking is the *application layer*, which establishes channels for computer processes to talk to each other. HTTP (HyperText Transport Protocol) is the most common application layer protocol, and establishes the rules for hypertext communication over the Web. Despite the name, HTTP isn't just for web content and is commonly used for IoT applications. Application layer protocols used in IoT are discussed in more detail in "IoT Application Protocols."

If a link layer protocol can support IP running on top, you can use it for any Internet communication. Ethernet, WiFi, and cellular data protocols can, but low-powered networks like ZigBee and Bluetooth cannot (yet). This is why many embedded devices cannot go directly online. (In "IP to the Edge," we'll talk about why this matters.)

## TYPES OF INTERNET NETWORK

Next, we'll look at some of the connectivity options that currently support Internet networking.

## Ethernet

If a wired connection is available and the device won't need to move around, Ethernet is a fast and reliable option. However, it's only good for consumer devices that don't move around, which tend to be kept in the home. Few people have a wired network at home, which means that the device will have to plug into one of the Ethernet ports on their broadband router. Using Ethernet runs the risk that the user won't have a spare port, and limits them to keeping the device near the router (see Figure 3-14).



**FIGURE 3-14**

Most people have only a few Ethernet ports at home, all of which are likely to be on their broadband router

## WiFi

WiFi can be used to create direct connections between devices, as well as Internet connections via a router. If a device has reliable mains power and tends to stay within the home, it's generally easiest to connect it via WiFi.<sup>5</sup> But be aware that within the home, some devices may be out of range of the network (e.g., in a garden, cellar, or just because the house is large).

<sup>5</sup> Although configuring WiFi on devices with limited user interfaces can be awkward. Chapter 12, "Supporting Key Interactions," contains detailed guidance on WiFi setup.

Providing customer support for WiFi devices can be complex, as home WiFi networks are often misconfigured. A misconfigured network may not pose an issue for a laptop but another device, such as a washing machine, might not be able to connect. Consumers may assume this is an issue with the appliance, but it might actually be an issue with the network itself.

### **Cellular data**

Cellular data uses the same data networks to which your mobile phone connects: usually GPRS or 3G/4G (see Figures 3-15 and 3-16). If the device needs to move around or you can't rely on the user connecting it to a WiFi network, using cellular data may make sense. It also requires minimal user setup: there's no need for the user to enter network details. But it uses a lot of power and can be expensive, requiring an ongoing subscription to a network provider. SigFox and Weightless are emerging alternatives to conventional mobile data. These have been designed for low power and data needs of the Internet of Things.



**FIGURE 3-15**

Glowcaps pill bottle sensors use a small gateway with its own cellular connection (image: Vitality)



**FIGURE 3-16**

Smart meters may use a GPRS radio to send data directly to the energy company's Internet service (image: Enexis)

## TYPES OF LOCAL NETWORKING

This section sets out the types of local networking most likely to be used for consumer applications. None of these currently support IP so cannot be used to connect devices directly to the Internet, only to a gateway. However, it is likely that in the future it will be possible to bring IP to edge devices over some of these radio types, giving them a direct Internet presence. (See “IP to the Edge” for more details.)

### Bluetooth

This is typically used to connect peripherals to mobile phones. Bluetooth is useful when the device is primarily used in range of a phone and in the presence of the user. This works well for wearables and detecting the presence of the owner (see Figure 3-17). It does not yet support IP networking in the real world but the standard allows for it.

Bluetooth LE (low energy; also known as Bluetooth 4 or Smart) is especially suited to connected devices with power constraints. Classic Bluetooth requires paired devices to maintain an open connection to each other, which means that their Bluetooth radios are constantly in use. This could drain the battery of a low-powered device very quickly.

Bluetooth LE allows for devices to connect and transmit data at regular intervals, or when they have something to share. This uses less power and preserves battery life.



**FIGURE 3-17**

The Sammy Screamer movement sensor from BleepBleeps sounds a loud alarm when out of Bluetooth range of its owner's mobile device; it could, for example, be used to detect when a bag is stolen (image: BleepBleeps)

Bluetooth LE is also beginning to appear in home gateways and home devices (see Figure 3-18).



**FIGURE 3-18**

The Dropcam Pro video camera incorporates Bluetooth LE for connecting to sensors and other devices in the home (image: Dropcam)

## Proprietary radio

Traditional baby monitors are an example of proprietary radio networks (see Figure 3-19). They use custom connections entirely controlled by the manufacturer. They may be reliable and cheap (there are no license fees for the manufacturer to pay as there would be with Bluetooth, ZigBee, or ZWave). But as the service provider, you have to have control of all the devices that are connected to them. They will not interoperate with other devices and may interfere with them.



**FIGURE 3-19**

Baby monitors often use proprietary radio networks to connect a baby listening base station with the parent unit

## ZigBee and ZWave

ZigBee and ZWave are low-powered radio networks, suitable for battery-powered devices and designed for use in home automation systems. Both have a similar range to WiFi. Unlike WiFi, the range of a ZigBee or ZWave network can be extended through *mesh networking*. This allows any of the devices on the network that run on mains power to act as *repeaters*, extending the range of the network. Neither currently supports IP networking.



The Nest thermostat connects to the Internet over WiFi, but also has a ZigBee radio. It could in the future act as a gateway for other low-powered devices in the home, giving Nest a route to becoming a home automation platform.

WiFi, Bluetooth, and ZigBee all operate on the 2.4GHz frequency, so there is a risk of interference. For example, there is a slight risk that, if you maxed out your WiFi network transferring videos, your ZigBee light switches might not work. In practice, problems are rare and WiFi routers are getting better at coexisting with other networks, but it's worth being aware of the risk.

### **RFID and NFC**

Radio frequency identification uses electromagnetic fields to transfer identifying data from small tags to a reader. Passive RFID tags are powered by an electromagnetic field from the reader. They have a very short range (10–100 cm). Active RFID tags have their own power source and can broadcast their ID over longer distances (10–100m). The microchip used to identify household pets is a passive RFID tag (see Figure 3-20), as is London's Oyster transport card.



**FIGURE 3-20**

The SureFlap pet flap reads the passive RFID tag in the pet's microchip and allows entry only to authorized pets (image: SureFlap)

Near field communication (NFC) is a set of communications standards built on top of RFID to enable two-way communication. It is designed for mobile phones and other low-powered devices. NFC is often used for mobile payment and works over a very short range (a few centimeters). NFC can also be used to simplify setup of more powerful network connections such as Bluetooth and WiFi. Android Beam and Samsung S Beam are examples of these. NFC could be used to make it easier for consumers to connect home devices together over other network types.

Chapter 12 covers design approaches for connecting devices together over different types of network. (Examples of RFID and NFC applications are also discussed in Chapter 2.)

### Powerline networking

Where Ethernet is unavailable and wireless networking too unreliable, it is possible to run data over preexisting electrical power lines using the HomePlug standard. An adapter, plugged into an electrical socket, is connected to Ethernet where a port is available (see Figure 3-21). Another adapter is plugged into an electrical socket near the device that needs connectivity, and connected to the device with an Ethernet cable. Data is transmitted over the electrical wiring.



**FIGURE 3-21**

A Devolo dLAN 1200+ HomePlug powerline adapter (image: Devolo)

In addition to being simple to set up, powerline networking is fast, secure, and reliable. It runs through thick concrete floors that might cause problems for WiFi. However, it acts as a radio transmitter and in some countries is legally classed as a form of broadcasting, which requires a license. Where multiple devices are plugged into it, it is possible for the signals to interfere with one another.

## IP TO THE EDGE

As we've seen, many edge devices cannot speak or understand Internet communications without translation by a gateway.

There is not yet a standard way to translate between Internet networking (TCP/IP) and most non-IP local network protocols such as Bluetooth and ZigBee. This means that custom translation must be built into the gateway apps.

This limits edge devices to doing only what the gateway apps allow. The Internet service can't talk directly to the edge devices to ask them to do anything else. For example, if you want to use your security motion sensors to inform your heating system that you are in, you have to change the apps on the gateway. The heating system can't just contact the motion sensors and ask them for the data.

But delivering Internet communications (over IP) all the way to the edge device, across even low-powered networks, is a growing area of interest, to provide many more edge devices with a unique identity on the Internet so they are able to contact (and be contacted by) any service and do anything they want. This is a true Internet of Things.

This would enable a far greater degree of flexibility of functionality. Instead of asking the gateway to pass on a message to a device and hoping that the gateway knows what to do with your particular request, you can contact it directly. You could compare this to the convenience of having a person's mobile number, as opposed to an office switchboard where you need to leave a message with a receptionist. If you have the mobile number, you don't need to know where they work in order to get hold of them.

Likewise, a device can identify itself uniquely, no matter how it connects or where from. For example, an electric vehicle could be charged at a friend's house, and the electricity billed to the car owner, not the friend whose house it was.

Giving all these connected devices unique IP addresses will require a change in standards. Most IP addresses are currently based on the standard IPv4, which provides 4 billion possible IPv4 addresses. This may seem a lot, but it's not enough to support the billions of connected devices predicted to come online in the next few years. The incoming standard IPv6 provides for the equivalent of 10 IP addresses for every atom on earth.

Initiatives are in development that would allow IP networking to run on top of Bluetooth, ZigBee, and other low-powered local area networks. 6LOWPAN is an IETF working group proposing solutions for IPv6 for low-power wireless personal area networks. In August 2014, Nest and other companies including ARM and Samsung announced the development of a new standard for low-powered networking, named Thread.<sup>6</sup> This can use existing ZigBee radios but supports IPv6.

When edge devices can speak IP, any standard higher-level Internet-based protocol can run on top. So you could use the same application protocol to talk to all your home devices. You no longer have to care about how the devices connect at the link layer, or translate between different network protocols. This will make it much easier for devices with different radios to interoperate. As engineer and founder of 1248.io Pilgrim Beart puts it:

Using the Internet (even to go a few metres from your smartphone to e.g. a PVR or audio system) may sound technically rather mad, but it has the huge benefit that it avoids the problem of physical standards, using the Internet as a lingua franca. Your phone and the PVR may not support the same physical standards but they can still communicate. It also allows you to remote-control from anywhere in the world.<sup>7</sup>

The standards and application protocols needed for this are still in development and are not yet consumer-ready. Also, making connected devices directly reachable on the Internet opens up security risks that would have to be mitigated.

Devices that use low-powered networking would still need to connect via a gateway with the appropriate radios to talk to those devices. But the gateway can be simpler: providing connectivity, a firewall, and local intelligence required to coordinate the devices when the Internet is unavailable. There is more potential to concentrate intelligence in the edge devices and the cloud.

---

<sup>6</sup> <http://threadgroup.org>

<sup>7</sup> In conversation

## Network Communication Patterns

Communication—how devices send and receive data to the network—can be handled in different ways. This is largely governed by protocols in the *application layer*. Understanding the way data flows around the devices in your service allows you to understand how responsive it may be and how rapidly data will be synchronized between devices.

### PUSH, PULL, AND POLLING

As a UX designer, the key factors that you'll need to be aware of are:

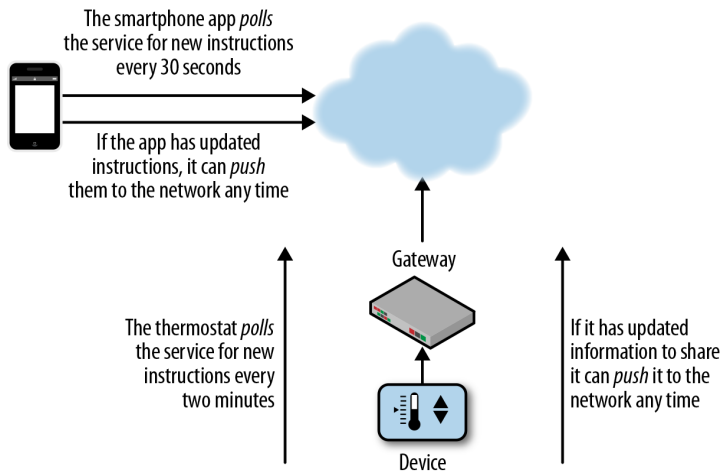
- How the device transmits messages *to* the network. Does it *push* them at regular intervals or when certain conditions are met? Or does it only transmit when a client device (like a smartphone app) or the service requests it (*pull*)?
- How the device receives messages *from* the network. Are messages pushed from the service or another device, or does the first device request them as needed?
- How frequently the device is connected. Is it constantly connected, connected at regular intervals, or only connected when it knows it needs to be?
- How fast instructions or data may get through. This is a combination of:
  - How long it takes the device to establish a connection
  - How fast the connection may be
  - How much data is being sent
  - How directly it may be routed (anything that goes over the Internet may take longer)

The following example, illustrated in Figure 3-22, illustrates some of the issues you may encounter:

A battery-powered heating controller using ZigBee contacts its gateway every two minutes to register its status and look for new instructions. This is called *polling*: the controller does not know when there are new instructions, but must check in regularly for updates.

The user turns the heating up from 19° C to 21° C using a mobile app. The app passes the instruction to the Internet service. The gateway device polls the Internet service at regular intervals and receives the

instruction. (The gateway has mains power, so it can poll quite frequently, perhaps every few seconds.) The heating controller polls the gateway and receives the instruction.



**FIGURE 3-22**

Polling and push in a heating system

If the user is in front of the heating controller while using the mobile app, they might see the controller take up to two minutes to respond to the command from the mobile device.

During this period, the phone UI and controller display will display different status information and will be out of sync. The phone may say 21° C, and the controller 19°C. The controller reflects the current true state of the system. In interusability terms, this is a *continuity* problem. (See Chapter 9 for more details.)

It would be quicker to push the instruction from the Internet service down to the gateway and device. But this is often not possible to do through home routers. Most routers present a single public IP address to the world and keep the addresses of devices on the local network private.<sup>8</sup> So it's difficult for servers to push to, or pull from, devices on local home networks. Security firewalls may also be an issue.

<sup>8</sup> This technique of sharing a single IP address is called NAT: network address translation.

So, the device has to poll out to check for new instructions, and this can take longer.

There's likely to be less delay if a user turns the heating up using the controls mounted on the heating controller device. The controller knows that there has been a change to the system, and pushes a message with the updated information via the hub to the Internet service. It knows it needs to connect so it doesn't wait for the next scheduled poll. If the user's heating control smartphone app is open, it might request new data from the service every 30 seconds. So if the user is looking at their heating smartphone app at the same time, they might still perceive a discontinuity, but only a short one.

Both the smartphone and the heating controller are polling the Internet service for changes, an example of *pull*. Both are able to *push* a message when they know they have updated information or commands to share.

Neither maintains a constant connection to the network, as this would drain battery power. But the phone can check in more frequently than the controller, as it is less power constrained.

Although this arrangement does expose occasional discontinuities, in the context of a heating system these are not disastrous. Most of the time, users are not standing in front of the heating controller with a smartphone. And most of the time, a two-minute lag between issuing the instruction to the heating and it actually turning on is not a problem: heating operates on a timescale of hours rather than seconds.

A polling interval that is too long can make the service feel too unresponsive. For example, using If This Then That (IFTTT),<sup>9</sup> a Philips Hue bulb can be configured to flash or change color when an email is delivered to your Gmail account from a particular recipient. That's a nice idea, but IFTTT is only able to poll Gmail every 15 minutes (perhaps a limitation imposed by Gmail to avoid undue load on their servers). This means the light may not respond until 15 minutes after the email was received. That's a very unresponsive notification system.

If you want a device to respond quickly to a command, it has to either maintain an open connection to listen for *pushed* commands, or poll more frequently. A good example of consumer push notification is

---

<sup>9</sup> <https://ifttt.com>

Google Contacts. If you update a Google Contact using the Web, an update is pushed to your phone within seconds. The phone is not regularly polling: it receives a spontaneous update notification from Google's servers.

Frequent polling when there is nothing new to share can place a load on servers (and exhaust batteries). Maintaining many open connections to lots of different devices is similarly impractical. But a device without an open connection cannot be reached for you to tell it to connect. When you need a device to respond quickly you may need an open connection. For example, the shutoff switch to your home's electricity supply should turn off right away if a gas sensor detects a leak.

It's easier to control the timing by which a device *sends* commands. A device can be configured to send data to the network when certain conditions are met; for example, the heating schedule is changed, or a moisture sensor detects water in a basement. It could be configured to keep trying to send data more frequently in certain conditions. If a heat detector that only transmits status every 2 minutes detects a possible fire, it might transmit every second for the next 20 minutes. If it could not contact its usual network gateway, it might hop through alternative nodes on a mesh network to try to get through. With a static IP address (e.g., IPv6) and the appropriate applications in place it could theoretically also try to use a neighbor's network in an emergency.

A device may also need ways to alert the user when a message has not gotten through. An elderly person's emergency alarm button should be as reliable as possible and offer backup connectivity, such as cellular data. But the button should still be able to tell the user not just whether their call for help was sent, but whether it was received by the system at the other end.

A key consideration for UX is to understand how quickly data is passed around the system, and whether this is a good fit for the context of use. For example, a home energy monitoring system for electricity usage might upload the last 24 hours' data to the Internet service every few minutes or longer. When you request this information, it would be sent from the Internet service rather than querying the energy monitor directly. This might be quicker, and/or reduce cellular data costs to your energy company. However, if you want to know whether your front door is locked, it's very important to know whether or not that data is 2 minutes old—someone might have opened it in the meantime.



## IOT APPLICATION PROTOCOLS

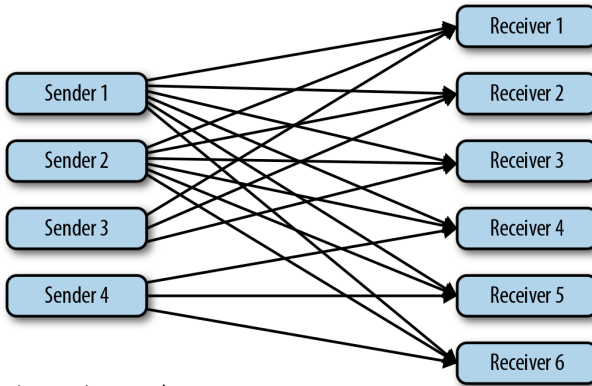
Application protocols sit in the topmost layer of Internet networking. They establish channels for computer processes to talk to each other. They shape the connection patterns by which devices pass and get data to and from the network. The choice of application protocol your system uses is largely a technical design issue, and is unlikely to have a big impact on UX. However, it's worth being aware of some of the options you may hear about, and understanding how the system you are working on passes data around.

It's common for IoT applications to communicate over the Internet using HyperText Transport Protocol (HTTP).

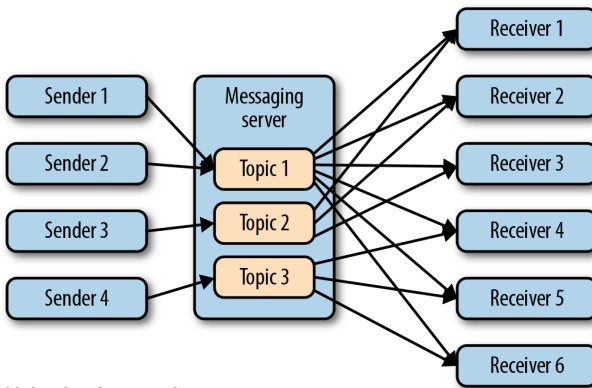
HTTP is a reliable way to pass information around a small number of devices that don't need to be updated instantly of any change. But it is not designed for push or streaming and so usually requires polling. It's also not the most streamlined method of passing data around lots of devices, especially ones that have power and computation constraints. It requires a direct connection to be established between any devices that need to share data (a point-to-point network, see Figure 3-23), and messages take up a lot of bytes.

Other protocols, notably MQTT, have been designed specifically for the Internet of Things. These prioritize smaller data transmissions and more efficient ways of passing data around large networks of low-powered devices. Instead of direct connections, MQTT uses a publish-subscribe model where devices push data to a central message broker. Other devices can subscribe to the data feeds ("topics") they want (see Figure 3-23). It can be fast: Facebook Messenger runs on MQTT. It's also a more flexible way to manage interconnections between lots of devices: each one simply tunes into the topics they want.

But HTTP is widely supported. It allows sensor networks to be explored in an easily accessible and universal way, through web APIs (discussed further in "APIs"). It also gets through firewalls, because it looks like normal web browsing.



A point-to-point network



A publish-subscribe network

**FIGURE 3-23**

Publish-subscribe application protocols, like MQTT, can provide a more efficient way to pass data round large networks of devices than point-to-point protocols (like HTTP). In this case, every receiver gets data from senders 1 and 2, but not all need data from senders 3 and 4. In the pub-sub network, fewer connections are needed to distribute messages correctly.

## Internet Service

In this section we'll look at the role of Internet services in IoT systems, and how the design of APIs (application programming interfaces) relates to UX design.

## THE ROLE OF THE INTERNET SERVICE

It's common for IoT systems to rely on some kind of remote centralized Internet service to collect, process, and distribute data and instructions.<sup>10</sup>

The Internet service would generally include some way of storing and handling:

- System data (e.g., sensor data or the current state of the devices)
- Applications that run on top (e.g., health monitoring or lighting)

The service makes data and system commands available to mobile or web apps via application programming interfaces (APIs). Increasingly, service providers make APIs available to third-party developers to enable other services to integrate with the system.

For example, data captured from a sensor network is aggregated and processed by an Internet server, which publishes a summary of the data.

System designers have a choice as to where processing happens: in the remote service (“cloud”), gateway, or edge device. Doing more processing in the cloud makes it easier to maintain one centralized view of the system. But the system won't work when devices lose connectivity and can't access the cloud.

That's acceptable for nonurgent data, such as air quality monitoring. If sensors go temporarily offline, they can store data and upload it when they reconnect. The worst that happens is that some data may be old.

But it's not OK where data must get through fast. It's unacceptable if a baby monitor does not alert a parent in the next room that the child has stopped breathing. This is where local processing is needed, so devices can work without the Internet (e.g., a thermostat with on-device controls, or a gateway that can continue to control the home lighting).

---

<sup>10</sup> A minority of connected devices may not have a remote centralized Internet service. For example, many WiFi printers have their own embedded web servers and thus host their own Internet service. But this is relatively uncommon in IoT.

An Internet service may be proprietary (custom built for your service), perhaps with APIs to share data with third-party services. Or it may be an open service (e.g., the Thingspeak<sup>11</sup> open data platform). If you are depending on someone else's service, then bear in mind that if they have an outage, raise their prices, lose or publicize your data, or go bust... then so do you.

A platform (in this context) is a type of software framework that can be used to build multiple remote service applications. Thingspeak, Kaa,<sup>12</sup> Xively,<sup>13</sup> and Thingworx<sup>14</sup> are examples of platforms that developers can use to create IoT services. Platforms can make it easier for engineers to get a system up and running, but might lock you in or lack flexibility. Relying on a third-party platform carries the risk that the platform provider may go out of business. Separate services connected with open standards may provide more control and choice: if you need to, you can swap out one service for an equivalent.

## APIs

An *application programming interface (API)* is an interface for developers. It provides hooks for them to build things out of your system. This might be your own frontend web or mobile apps or third-party services that interact with your system. APIs are an essential part of creating ecosystems of interoperable devices and services.

APIs enable developers to create web services that can easily be linked to other services. So you needn't build your own weather forecasting service for your rain-predicting device: you can easily call on a third-party weather service's API. The principle of "small pieces, loosely joined" encourages developers to create targeted services that do one thing well and can interoperate well with other services. This provides the flexibility to combine different services together to create entirely different applications.

---

11 <http://thingspeak.com>

12 <http://www.kaaproject.org/>

13 <https://xively.com/>

14 <http://www.thingworx.com/>

UX people are not usually responsible for designing APIs, but API design can have a significant impact on the UX you are able to create. The functions you make available in your APIs determine both what you can do with your own UX design, and what others (third parties) can do with your service.

Designing APIs is very much like designing an information architecture for a content website. You need to understand what information your users need, and how it should be chunked and organized to make it easy and quick for them to find what they want.

Sometimes design decisions will involve trade-offs: by making some tasks easier, you'll make others harder.

If your UX design and your API design are a poor fit for each other, the result can be long latencies, poor responsiveness, and overloaded servers. API design should run hand-in-hand with product definition and UX design.

### **How do web APIs work?**

APIs can describe data or function calls. They can be written in many languages, but in this chapter we'll use the example of standard web APIs, which are based on HTTP.

Web API calls apply actions to URLs. Here's an example of a possible API URL structure for a connected home system: *http://api.[service providername].com/property/gateway/devicetype/device/channel*.

- The first part is the service provider's public domain name.
- *property* is the home in which the system is based (e.g., Claire's house).
- *gateway* is the name of a gateway device (in case you have more than one).
- *devicetype* is the name of a type of device (e.g., motion sensor, contact sensor, light switch).
- *device* is the name of a specific device assigned by the user (e.g., dining room motion sensor or back door contact sensor).
- *channel* is a data feed from that device. A device might have more than one (e.g., all motion and contact sensors might also sense temperature, so would have two feeds: motion and temperature).

You can use the API to retrieve data, or in some cases, send commands to the system, by sending requests using HTTP *methods*.

The GET method is the most common and is used to retrieve data. To use our example: if you wanted to retrieve status information from all the motion sensors in the house, you could do so by making a request using the GET method to: `http://api.[serviceprovidername].com/clareshouse/gateway1/motionsensors`.

The system would respond with something like the following XML code:

```
<channel type="motion">
  <events>
    <event type="motion" detected="no" timestamp=""
      location=""/>
    <event type="motion" detected="no" timestamp=""
      location=""/>
    ... (one entry for every motion sensor in the house)...
  </events>
</channel>
```

If you have the authorization to do so, it's also possible to use other methods to modify data on the server.

In our example, if you wanted to turn on the light in the dining room to 50% dimmed, you could make a PUT request to:

```
/{ clareshouse/gateway1/lightswitches/diningroom
<light>
  <state value="on"/>
  <dimming percentage="25"
</light>
```

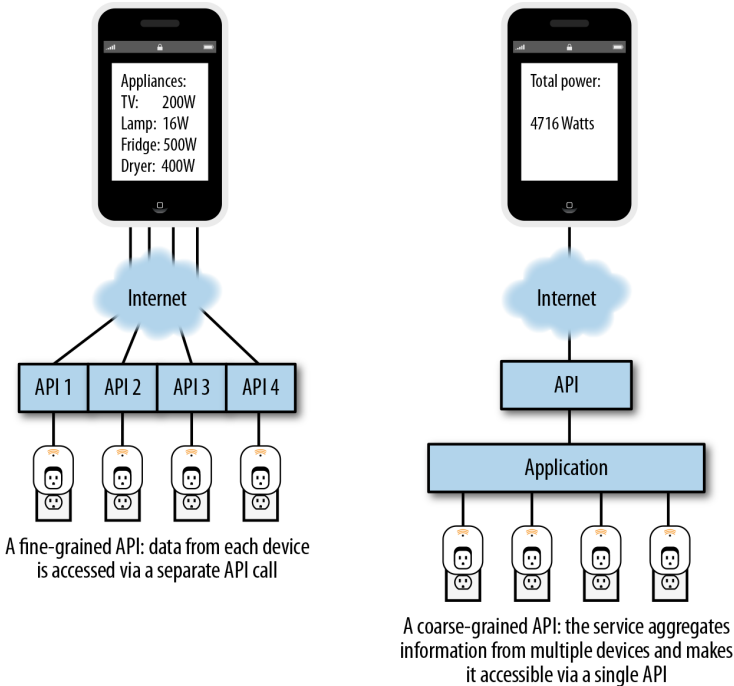
These are very simple examples, but should help you understand the design issues described in the following sections.

### Key design issues for APIs

The key decisions to be made when designing APIs are the granularity of data and controls, and how you organize them into hierarchies.

**Granularity.** A simple system, composed of one device connected to the Internet, can have a simple API. The device posts key data it has collected and you can pass simple commands (controls) back to it, as in our previous example. This is a *fine-grained* (or *end point-oriented*) design, as it exposes low-level data from the device itself.

End point/fine-grained APIs (Figure 3-24) allow for control of specific devices or retrieving specific pieces of data, which is fine for small systems. But if you want to retrieve data from many devices at once, a fine-grained system means making lots of individual API calls. This will be slow: each API call could take 1 second per round trip on a 3G connection. A web page or mobile app screen that has to make 20 API calls will be much slower to load than one that only makes two or three.



**FIGURE 3-24**

Simple examples of fine-grained and coarse-grained APIs

In IoT systems with lots of devices, the individual devices often aren't important to the user anymore. If you want to know the temperature of your living room, you want to think about the room, not about the devices in that room that can sense temperature. The system should calculate the room temperature for you and make this available via a single API call.

These are examples of *task-oriented* (designed for specific tasks) or *coarse-grained* (aggregated data) API functions. The service aggregates data from multiple devices and applies some processing on the back-end. Your smartphone app then only has to make one API call to know which lights in the house are on or off.

Task-oriented APIs make it easier to support complex functions across multiple devices. You could tie your API tightly to your UI, aggregating all the data needed for a particular screen or widget accessible via a single API call. This would be highly responsive but would need re-designing if you ever wanted to change your UI. Any third parties using your APIs would have to present data in exactly the same screen/widget chunks as you. So task-oriented APIs risk inflexibility: UX designers and developers have to use them in the ways that you bake into the API design.

If an API is too coarse grained, it may also prevent you from accessing the individual device or data you need. You may sometimes want to turn off all the lights, but you also sometimes want to turn off just the hallway light.

Niall Murphy from Evrythng cites the real-world example of a connected car with a single API. This is designed for repair/manufacturer diagnostics, but impractical for other uses:

You can either turn on a stream of all the information from the car or turn it off. The home doesn't need a stream of the location of the car at all times to open the garage door when you get home. You need to bite-size info down to what is needed for a specific application.

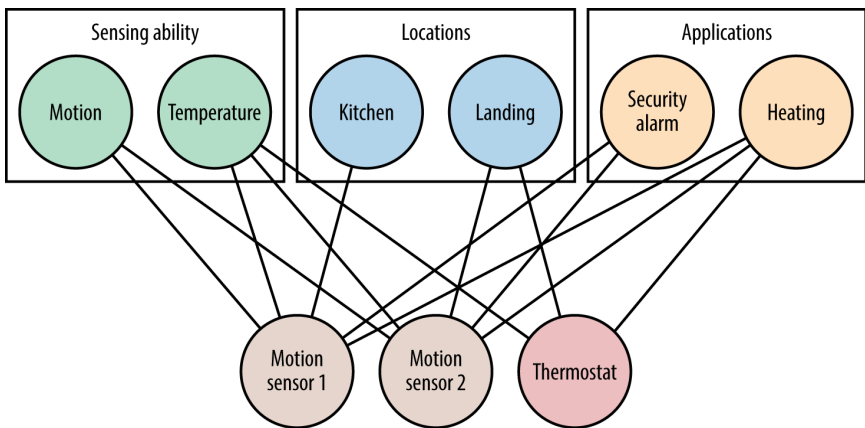
It's possible to have a mixed design, with some aggregated data and task-oriented APIs, and some end point (device) APIs. This would enable you to show the energy usage of all the appliances in the home, but also just to track usage of the tumble dryer.

**Structure.** As well as choosing the right granularity of data and controls, you also need to group and organize units (devices and data points) in an API to make sense/be usable to others. This becomes increasingly important the more devices and data points you have.



The way in which data/devices are organized in your APIs can make it easier or harder to retrieve data or apply controls from/to the combination of devices you may want. Trying to create a UX design from an API design that is not a good fit will result in a sluggish, unresponsive UX.

Devices in a home might be organized by their sensing ability (e.g., temperature or motion), location (e.g., kitchen or upstairs), the applications they support (e.g., security or heating), and more. Some can do more than one thing—for example, some devices can sense both temperature and motion and/or might be used by more than one application. If you drew a map of the interrelationships between those devices, it would look like a network structure (see Figure 3-25).



**FIGURE 3-25**

Interrelationships between devices form a network structure

Users might want to access different organizational groups. For example, you might want to see temperature readings around the house. Or you might want to view all the security devices. Or you might want to check up on the current status of every device in the living room.

For complex, multidevice systems, a network type UX often seems like an intuitive way for the user to explore what's available. From the living room motion sensor, you could browse to other security devices, other motion sensors, and other devices that measure temperature.

The trouble is, providing this much flexibility in the UX isn't necessarily easy using APIs. In particular, looking up *related* data is often a challenge. URLs are hierarchical, so they don't reflect all the ways data can be interrelated.

Take the dining room temperature example we looked at previously. On the same screen as the dining room temperature, you also want to show readings from other temperature sensors in the home. To the user, this might seem like an obvious thing to do: from one temperature reading, I can view the others. But in this schema, they all sit at the bottom of the tree structure and their relationship to one another is not mapped. So you'd have to call the data from each of the other temperature sensors individually. This would be slow and inefficient, just as if you were having to browse up and down the tree structure of a website to compare lots of individual pages of content.

If you're designing a small system with only a few devices, then the potential inefficiencies are small. But the more data points and devices you have to contend with, the more important it is that your API structure is a good fit for the UX.

As a UX designer, the important thing is to make sure that you work closely with your developers as the APIs are designed.

**Third-party APIs.** You may be relying on a third-party API, such as Xively or Twitter. If so, you're bound by their terms and you need to check how you can use them. Providers often impose limitations on the frequency of calls you can make, or the number of API calls per day to avoid overloading their own servers. They may also charge for API calls, as the weather service [forecast.io](http://forecast.io) does.<sup>15</sup>

## Summary

Networking has a fundamental impact on the UX of an IoT system. Many connected devices connect only intermittently to the network to poll for new instructions. This can create small delays in which parts of the system are out of sync with each other. Network latency and reliability issues may mean that connected devices don't behave as we expect real-world objects to do.

The system architecture determines how devices pass data around a system, where processing happens, and what will or will not work when parts of the system are offline. Devices may connect directly to the Internet or via a local connection to a gateway. Many devices can only

---

<sup>15</sup> And, as with third-party platform providers, they may go out of business.

support low-powered networking like Bluetooth and ZigBee. Gateways enable these devices to be connected indirectly to the Internet. In the future, delivering IP over low-powered networks will enable more devices to be directly connected to the Internet. This use of Internet standards will make it easier for different systems to work together.

Application protocols shape the connection patterns by which devices pass and get data to and from networks. Data may be pushed or pulled to a device; the device may be constantly connected or poll for instructions at regular intervals. HTTP is commonly used; specialist IoT protocols like MQTT are optimized for low-power devices with limited bandwidth.

All IoT systems rely on some kind of remote centralized Internet service to collect, process, and distribute data and instructions. The service makes data and system commands available to mobile or web apps via application programming interfaces.

An *application programming interface (API)* is an interface for developers. It provides hooks for them to use system data to make end user applications and in other systems. If the API design is a poor fit for the requirements of the UX, it can result in a slow, unresponsive experience or overly restrict the functionality you can offer.



## [Case Study 1]

# Proteus Digital Health: The Connected Pill

**BY ARNA IONESCU**

Medicines are an amazing innovation of the 20th century, but they only work when taken correctly. Experts estimate that half of medication is not taken as prescribed, meaning missed doses, wrong doses, or wrong medications.<sup>1</sup> Due to these adherence challenges, the majority of people fail to get maximum value from their medication. Proteus Digital Health is addressing this problem by adding sensors to oral medications and connecting them to the Internet of Things.

These connected medications are providing patients, caregivers, and healthcare professionals with new data streams. Proteus' system unobtrusively captures exactly when people take their pills, and places that data alongside other physiologic and behavioral data such as heart rate, activity, and sleep patterns. Together, this information enables new methods of self-management and generates new insights to inform clinical decisions.

## The Technology

The system's core innovation is a tiny and low-cost 1mm × 1mm ingestible sensor (see Figure CS1-1) that can be placed into pills during the manufacturing process. This disposable sensor is designed to be fundamentally safe, made from elements typically found in the human diet. It is also invisible to the user, enclosed completely inside the pill, so the "connected" pill looks no different from a standard one.

---

<sup>1</sup> World Health Organization, "Adherence to Long-term Therapies: Evidence for Action." Accessed December 2014, <http://bit.ly/1F68Jra>.

When designing the physical sensor, we felt it was important to maintain familiarity with the medication people are accustomed to taking, rejecting attachment methods that left the sensor visible.



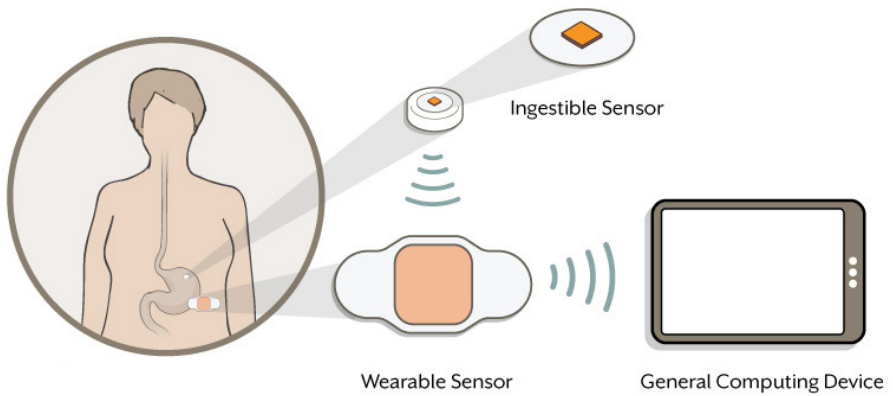
**FIGURE CS1-1**

The Proteus ingestible sensor

The ingestible sensor communicates via a proprietary transmission technology to a small wearable sensor, developed by Proteus and contained within a small patch that is worn on the torso and replaced weekly. The wearable collects additional data such as activity and heart rate, and transmits everything to a mobile phone (see Figure CS1-2). The phone then sends the data to the cloud, where it can be analyzed and accessed via secure web portals and apps. The apps can be tailored to different use cases and optimized for different users. People have different uses for the information: patients manage daily care, personal caregivers gain reassurance about their loved ones, nurses triage who needs immediate attention, doctors inform their clinical decisions, and healthcare administrators dig into their population's trends.

It was also important to design a service that enabled the technology to launch successfully. We initially had two service design goals: (1) lower barriers to using an unfamiliar product, and (2) accelerate our learning about how we can improve our product. To address these goals, we invested in hiring fulltime “coaches” who were responsible for visiting early customers in their homes to help them use the system, and also to report back about their challenges. This investment has been critical, enabling us to quickly identify and fix product pain points. As the

product becomes easier to use and customers become more confident with the new technology, we expect to shift to a more cost-effective and less resource-intensive service model.

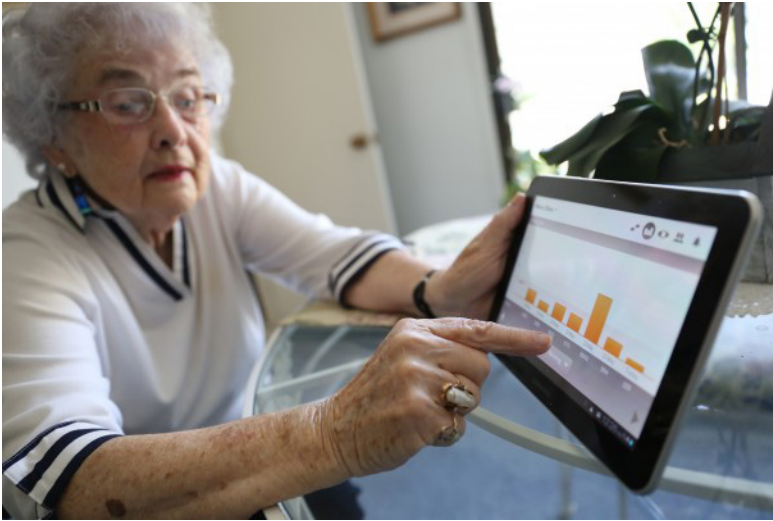


**FIGURE CS1-2**

The Proteus system

One of our early design tasks was to figure out who needs to access what data and how. The data generated by Proteus can be used in a variety of contexts for myriad reasons. It was key for us to narrow in and focus on just a few use cases to start.

Our business team identified a set of potential markets. Then, to define the product offering, our design research team visited potential customers in those markets in their homes or clinics and carefully watched for their deepest challenges, listening for the specific questions that Proteus could directly answer. Using these learnings, our interaction designers created potential information visualizations in the form of static screenshots that we brought back into the field for feedback (see Figure CS1-3). While our early prototypes displayed lots of data—exhibiting the breadth of the system’s capabilities—the feedback we gathered drove us to instead implement information visualizations focused more directly on the questions asked, leaving a large amount of the data unused. We found that providing too much data generated noise that was counterproductive to helping people maximize their benefits from using the system, especially in the early stages of establishing a “culture of use” with the new technology.



**FIGURE CS1-3**

Testing visualizations with users

## Early Use Case 1: Caregiving

The technology is well suited to help caregivers stay abreast of fragile family members from afar. We met with families and listened to their stories. The main need we heard from caregivers was knowing whether mom took her morning medications and made her usual sandwich at 2 p.m. Mom's main need was to maintain dignity. Our information visualization for this use case involved a simple ribbon that leveraged medication-taking and activity data, allowing a family to see a high-level overview of a loved one's pattern throughout the day (see Figure CS1-4). So, for example, a family would know that mom was up and about at the time when she makes her usual sandwich. The visualization avoided clinical data such as heart rate that did not provide immediately meaningful insight to family members and had the potential to invade mom's privacy in this personal use case.





**FIGURE CS1-4**

The tablet app showing the ribbon.

## Early Use Case 2: Informing Hypertension Treatment

The second use case involved helping physicians determine why hypertension patients presented with uncontrolled symptoms. Doctors have little insight into whether a patient is uncontrolled due to not taking medication or due to being prescribed the wrong medication. After using the Proteus system for two weeks, the physician can access a succinct report that answers two key questions: did the patient take the medication, and is the condition now under control? Activity data was not included in the report, as it was not useful in answering the physician's key question. Most patients will take medication as prescribed during the observation period due to the Hawthorne effect—the awareness of being observed changes a person's behaviors for a short duration of time—so the clinician now has verified adherence and an indication of whether the drugs are working when taken. This combination provides actionable insight that helps a clinician decide between adherence counseling or escalation to the next line of therapy.

## Discussion

The focused use cases were critical in building early trust within the market. It is difficult for people to immediately accept new technology, especially technology made intimate by the fact that it is swallowed. We fostered trust by demonstrating to the market that we were striving to understand our customers and directly address their needs, without clutter and unnecessary invasiveness. We were solving real problems for people.

Additionally, the early use cases were forgiving to potential faults that a new technology might experience when first released. Fault resilience was a critical criterion in selecting where to start. The system uses a standard mobile phone to transmit data, and we knew the phone would sometimes be apart from the user (so unable to gather data from the wearable), out of range (so unable to transmit data to the cloud), or off (neither gathering nor transmitting data). Additionally, while we were able to shift to continuously flowing data from the wearable to the phone when Bluetooth Smart became available (with standard Bluetooth we could only transmit data periodically), there is currently no way to have a continuous data connection from the phone to the cloud while maintaining a full day of battery use on the phone.

To mitigate, we designed our wearable to store an entire week's worth of data, in case the phone was never in range. We also designed a smart data push system, so that connectivity between the phone and cloud was triggered by key events, such as medication detection, making the system feel more continuous than it really was. Most importantly, we consciously looked for use cases that did not require highly accurate in-the-moment information—both use cases cited earlier are focused on trends. We built trust by launching a product we could guarantee works as expected, rather than launch a product with a high risk for reliability issues. As continuous data connections become more robust, we expect to have a host of additional use cases to explore.

Trust also comes from immediately establishing the right mental model about a new technology. When hearing about a new product, people often form immediate misconceptions. We had to get ahead and help people quickly establish the right models. The ingestible sensor, for example, uses a proprietary transmission technology that only works with the wearable provided by Proteus. This makes the system inherently secure and private, but a common misperception is that

people will be emanating signals that anyone can pick up whenever they take their connected medication. We helped people understand from the first moment of interaction how the system is fundamentally designed for privacy, explaining not only our use of a proprietary transmission technology but also the advanced security protocols used by the databases and applications.

Proteus is passionate about solving the difficult problem of helping people maximize the benefits of their medications. A broad range of expertise is required to address such a difficult problem as well as to create a connected product. Connected products are not as simple as a standalone app or a novel piece of hardware. They are all that, and they have to integrate seamlessly into the lives we live.

At Proteus, the complex design team includes UX/interaction/UI designers, industrial designers, graphic/visual designers, packaging designers, and a design research team. Every designer on the team is professionally “T-shape”—meaning great skill and depth in one area of expertise, as well as breadth across design disciplines, allowing close collaboration with fellow teammates. The team is also deeply focused on collaborating across the organization. First, designs without champions in the product and engineering organizations are not particularly useful, and second, with such a complex product that contains many parts, everyone involved needs to be passionate about serving the needs of our customers.

In order to create truly groundbreaking IoT products, we embedded a sensitivity for users throughout the organization. We nurtured user champions outside the design team, training anyone interested in how to conduct user observations and bringing them along on field visits. We also educated the organization widely, ensuring that all of our collaborators knew how our team could help. This focus on collaboration, both within the design team and more broadly, is a critical component to creating complex, new-to-the-world products that have a chance at successful market adoption.



# Product/Service Definition and Strategy

**BY CLAIRE ROWLAND**

We all aspire to create the killer product or service that people want to buy and love using. The key to this is ensuring that the product solves an actual problem that people have, in a way that appeals to them. At a pinch, it might provide them with something new and wonderful that they never knew they needed. It sounds simple and obvious, but it can be remarkably difficult to get this right. Right now, the IoT market is skewed toward innovators and early adopters. There's huge potential to create great new products for consumers, but they may have to contend with new types of complexity.

This chapter introduces:

- Productization as part of IoT design (see page 112)
- Moving from innovation to mass-market products (see page 116)
- How products differ from tools (see page 123)
- What makes a good product (see page 128)
- Building service offerings around products (see page 135)
- Business models in IoT (see page 144)

This chapter addresses the following issues:

- Why a clear value proposition is a prerequisite to great UX design (see page 113)
- The different types of market for consumer IoT products (see page 120)
- Why consumers want products, not tools (see page 123)

- Why it's important to design the service offering around a product (see page 143)
- How business models can shape UX (see page 145)
- How digital business models may start to appear in real-world products (see page 147)

## Making Good Products

In this section, we'll look at why creating a compelling product proposition is the necessary foundation for a good UX, and why it's a particular challenge in IoT right now. We'll also consider how an IoT product need not be a physical device: it can also be partly (or completely) a service.

### WHAT IS PRODUCTIZATION?

*Productization* is the extent to which the supplier makes the user value of the product explicit and easy to understand. Compelling products don't just look good or otherwise fuel some underlying need for status (although those things are often important). They make it immediately apparent to their intended audience that they do a thing of real value for them: preferably something new that serves a previously unmet need.

Nest is probably the most famous IoT productization success story. Consumers were resigned to thermostats and smoke alarms being ugly, annoying boxes with usability flaws. It hadn't occurred to most people that they could be better. Nest products promise to do the job better than most of the competition, in the form of attractive and desirable hardware that users are happy to have on show at home (see Figure 4-1). Of course, they are premium products with a premium price tag. The point here is not that all products should be expensive, but that a good product should fulfill a clear need for the target audience, with a usable and appealing design. *This is the product's value proposition: the user's understanding of what the product does for them and why they might want it.*

As Denise Wilton, designer (and former creative director of design agency BERG), puts it:

Never underestimate the power of a simple explanation, or a product that looks nice. If people can understand it, they can want one for themselves. They're not scared of it. It stops being a weird thing that geeks do.<sup>1</sup>



**FIGURE 4-1**  
Nest thermostat shown in home (image: Nest)

### **WHY IS THIS IN A UX BOOK?**

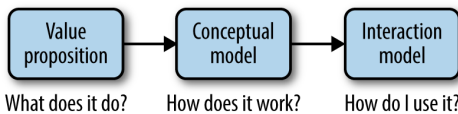
To some of you, this may seem outside the remit you normally associate with UX design. You may work in a company where productization is handled by product management, or perhaps marketing. In others, it might be considered strategic design. UX is not always involved in identifying the opportunity and framing the solution. But most UX designers would walk over hot coals to be involved from the start, especially if they have firsthand knowledge of user needs from conducting research.

Whoever is responsible for it in your organization, it provides the strategic foundation for UX design. *It's not possible to design a great product or service experience if users don't want, or understand, the service in the first place.*

---

<sup>1</sup> From a talk at UX Brighton, November 2012.

Value propositions help sell products. But they also drive UX. A clear proposition helps users decide whether to buy it in the first place, but also helps frame their mental model of the system and what it does (see Figure 4-2). When users are confident that they understand what the system does for them, they have a good basis for figuring out how it works (the conceptual model), and then how to use it (the interaction model). All the clever design in the world can't overcome a murky or unappealing value proposition.



**FIGURE 4-2**

A good clear value proposition is fundamental to a great UX

### **WHY IS THIS IN AN IOT BOOK?**

Productization is, of course, not a challenge that is unique to IoT. It is included in this book as it is a particular challenge for the consumer IoT field right now. Many products and services aren't yet offering good, practical solutions for proven consumer problems. Even where they are, the value isn't always apparent from the product itself or clearly stated in terms target users would understand.

This isn't a criticism of the many clever and talented people working in this field. Most of them are aware that consumer experience is a challenge.

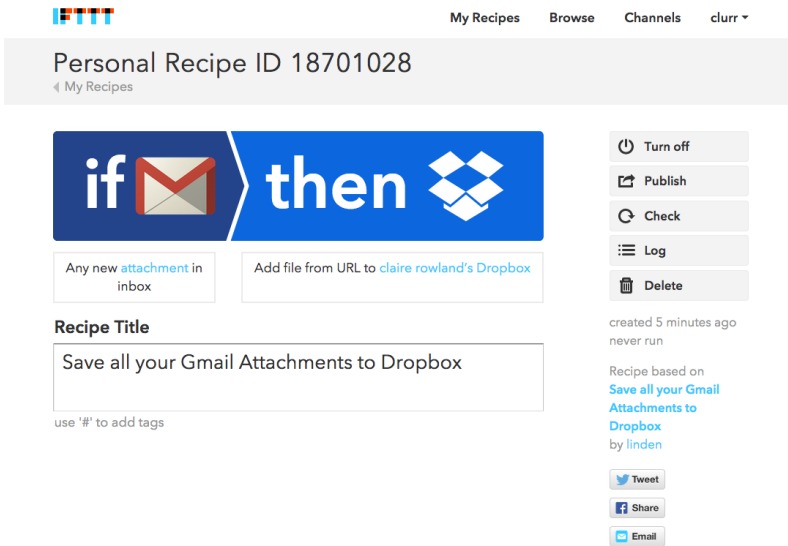
It's a result of the novelty and inherent complexity of the products and services. We're still figuring out what we can do with the technology, and we're asking users to wrap their brains around some novel devices and capabilities.

It also reflects that new technology products and services are often conceived and developed by people with an engineering mindset who value highly configurable functionality. These initiatives can often seem complex and unclear in purpose to consumers, because in trying to do so much, they fail to communicate a clear value for using the service.

There is, of course, a market for products developed to meet the needs of highly technical users. There's also great value in products and services that help a wider range of people move beyond passive consumption of



technology and learn how to construct their own solutions. For example, If This Then That offers an accessible way to coordinate different web services and even connected devices (see Figure 4-3). This is functionality that would previously only have been available to those with good programming skills.



**FIGURE 4-3**

An If This Then That recipe for saving Gmail attachments to Dropbox

## PRODUCTS CAN BE SERVICES

When we talk about IoT, we tend to focus on the edge devices: the activity monitors, thermostats, connected pet feeders, and more. This is especially true when the devices themselves look novel (such as the Nabaztag rabbit shown in Chapter 2) or striking (such as the Nest thermostat).

But while the devices are a key part of the UX, they are not the whole picture. They are all dependent on an Internet service. This makes the user's relationship with the product much more dynamic. Instead of the traditional one-off purchase of a traditional physical product, the user interacts with the provider on an ongoing basis. The user's experience isn't just shaped by the device—it's shaped by the whole service. There might not even be a physical product at all: just as you can now pay for Dropbox storage or personal fitness training, so you may pay for

software or storage to help you make the most of connected devices, or personalized health or energy-saving advice based around data gathered from your devices.

### [ NOTE ]

In this book, we use the term “product” loosely to refer to a packaged set of functionalities that solves a problem for people or fits neatly into their lives. That could be a physical device, a service, or frequently a combination of both.

But the bigger challenge is in creating products and services that work for mass-market consumers. For this audience, the functionality—what the system does and how to use it—should be transparent. The underlying technology should be invisible. The user should be able to focus on getting the benefit from the product that they were promised, not on configuring it and maintaining it.

## From Innovation to Mass Market

The primary focus of this book is on creating consumer IoT products and services. In this section, we take a brief look at how technological innovations cross over into the mass market and consider what lessons there may be in here for IoT.

### INNOVATORS ARE NOT CONSUMERS

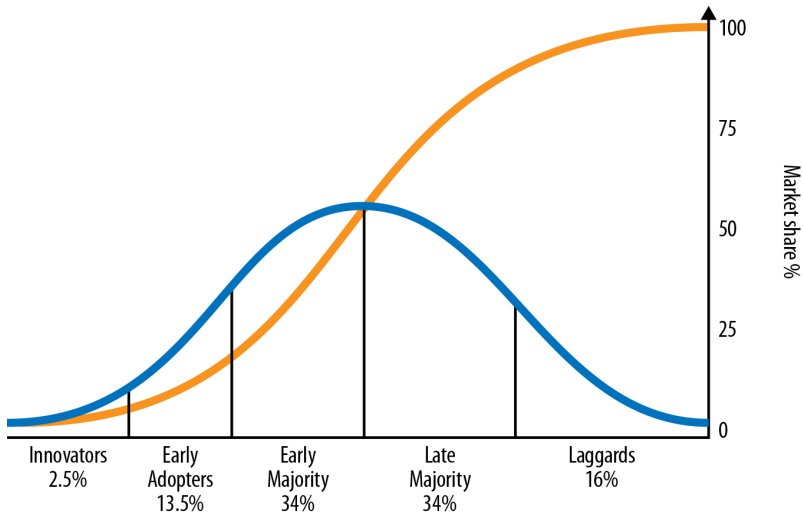
In 1962, the sociologist Everett Rogers introduced the idea of the technology lifecycle adoption curve, based on studies in agriculture.<sup>2</sup> Rogers proposed that technologies are adopted in successive phases by different audience groups, based on a bell curve (see Figure 4-4). This theory has gained wide traction in the technology industry. Successive thinkers have built upon it, such as the organizational consultant Geoffrey Moore in his book *Crossing the Chasm*.<sup>3</sup>

---

2 Everett M. Rogers, *Diffusion of Innovations*, Fifth Edition (New York: Simon & Schuster, 2003).

3 Geoffrey Moore, *Crossing the Chasm* (New York: HarperBusiness, 1991).

In Rogers's model, the early market for a product is composed of innovators (or technology enthusiasts) and early adopters. These people are inherently interested in the technology and willing to invest a lot of effort in getting the product to work for them. Innovators, especially, may be willing to accept a product with flaws as long as it represents a significant or interesting new idea.



**FIGURE 4-4**

The diffusion of innovations according to Everett Rogers; the blue line represents the successive groups adopting the technology, the yellow line the market share (image: redrawn from Tungsten's image on Wikicommons)

The next two groups—the early and late majority—represent the mainstream market. Early majority users may take a chance on a new product if they have seen it used successfully by others whom they know personally. Late majority users are skeptical and will adopt a product only after seeing that the majority of other people are already doing so. Both groups are primarily interested in what the product can do for them, unwilling to invest significant time or effort in getting it to work, and intolerant of flaws. Different individuals can be in different groups for different types of product. A consumer could be an early adopter of video game consoles, but a late majority customer for microwave ovens.

Geoffrey Moore identified a “chasm” between the early adopter and early majority market (which he called visionaries and pragmatists). These groups have different needs and different buying habits.

Mainstream customers don't buy products for the same reasons as early adopters. They don't perceive early adopters as having the same needs as themselves. Mainstream customers may be aware that early adopters are using the product. But this will not convince them to try it out themselves unless they see it as meeting their own, different, needs. So products can be successful with an early market, yet fail to find a mainstream audience.

An example of this in the IoT space is the home automation market. Systems such as those based on the power line protocol X10 have been around for close to 40 years. (Early examples ran over electrical power lines and analog phone lines.) The example in Figure 4-5, from 1986, shows a system that allowed users to program and remotely control their heating, lighting, and appliances over a (landline) phone. These are all applications that still seem novel and innovative to us; they would have excited the innovators of the 1980s even more.

However, home automation remained a niche market. It was expensive. It required significant technical skill to set up and maintain. Even those mainstream consumers who had heard of home automation did not see much value in programming their heating, lighting, and appliances. Had it been more affordable or easier to use, more people might have been willing to try it out. But only now are consumers starting to see the utility of connected home products. This is arguably driven by the rise of the smartphone, giving us a metaphor for the “remote control for your life.”

## **WHAT'S DIFFERENT ABOUT CONSUMERS?**

Mainstream consumers are now more aware of connected devices, but they need to be convinced that these products will actually do something valuable for them. A product that appeals to an audience that loves technology for its own sake cannot simply be made easier to use or better looking. To appeal to a mass-market audience, it may need to serve a different set of needs with a different value proposition. Chapter 5 covers learning about user needs and some of the special considerations you might encounter when designing for IoT.

**THE X-10 POWERHOUSE INTERFACES WITH YOUR COMMODORE TO CONTROL YOUR HOME...FOR SECURITY, COMFORT AND ENERGY SAVINGS.**

This remarkable Interface lets you run your home through your Commodore 64 or 128 and a keyboard or joystick.

When you're away, it makes your home look and sound lived in. When you're home, it can turn off the TV at night and wake you up to stereo and fresh brewed coffee in the morning. It can even turn on your air conditioner and control your heating.

**SPECIAL COLOR GRAPHICS MAKE PROGRAMMING A SNAP.**

You simply pick a room from the display screen. Use your keyboard or joystick to position graphics of lights or appliances. Then follow on-screen instructions to program any light or appliance to go on or off whenever you choose. You can even control thermostats, light intensity and more.



**THE WAY IT WORKS.** The X-10 Powerhouse Interface is cable-connected to the Commodore "User" port and plugged into a standard 110V outlet. After it is programmed, the Interface sends digitally encoded signals through your home wiring to special X-10 Modules. To control a lamp or appliance, you simply plug the electrical device into a Module

**THE X-10 POWERHOUSE DOES EVERYTHING BUT PUT OUT THE CAT.**



Model CP290



and then plug the Module into an outlet. The Interface can control up to 256 Modules throughout your home and won't interfere with normal use of lights and appliances.

There are plug-in Appliance Modules, Lamp Modules, Wall Switch Replacement Modules and Special 220V Modules for heavy duty appliances such as water

heaters and room air conditioners. Plus Thermostat Controllers for central heating and air conditioning, Telephone Responders to control your home from any phone, and much more.

**IT WON'T TIE UP YOUR COMPUTER.** Use your computer only for programming. When you're finished, disconnect the Interface from the "User" or RS-232 port and keep it plugged into any convenient power outlet in your home. It will operate as a stand-alone controller with battery back-up and will run your home automatically.

**SURPRISINGLY INEXPENSIVE.** A Powerhouse System including the Interface, software and connecting cables costs less than \$150. X-10 Modules are less than \$20 each.

For the Dealer Nearest You Call: **1-800 526-0027**  
[In NJ: (201) 784-9700]

or, write to: X-10 (USA)  
185A Legrand Avenue  
Northvale, NJ 07647  
[www.commodore.ca](http://www.commodore.ca)  
**X-10 POWERHOUSE**  
NUMBER ONE IN HOME CONTROL

Commodore 64 and Commodore 128 are registered trademarks of Commodore Int'l Ltd.

**FIGURE 4-5**

Advertisement for X10 Powerhouse for the Commodore 64, from the January 1986 edition of Compute! Magazine (image via [www.commodore.ca](http://www.commodore.ca))

Mass-market product propositions have to spell out the value very clearly. Users will be subconsciously trying to estimate the benefit they'd get from your product as offset by the cost/effort involved in acquiring, setting up, and using it, and you need to be realistic about the amount of effort they will be prepared to invest in your product. The further along the curve they are, the more users need products

with a clear and specific value proposition, which require little effort to understand or use. And they have a very low tolerance for unreliability. Your product has made a promise to do something for them, and it must deliver on that promise.

This is not simply a question of lacking technical knowledge, and certainly not of users being dumb. That 10-step configuration process to set the heating schedule might seem trivial in the context of your single product. But it can feel overwhelmingly complex in the context of a busy life with many other more pressing concerns. For this reason, consumers tend to be most attracted by products that seem as if they will fit into their existing patterns of behavior and don't require extra effort. For example, ATM cards and mobile phones were arguably successful because they reduced the need to plan ahead in daily activities (getting cash from the bank, or arranging to meet).

### **VALUE PROPOSITIONS FOR IOT**

The guidelines just outlined can of course be applied to any type of product or service. But connected products can be complex and often do novel things that are hard to communicate succinctly.

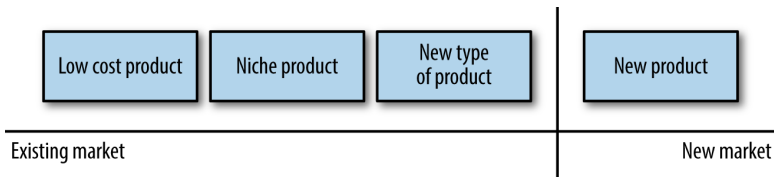
Core value propositions should be straightforward—for example, a company offering smart meters may promise to “tell you where your energy spend is going,” which is relatively simple. A good test of an IoT product proposition is that end users should not need to focus on its connectivity or onboard computing: it should just make sense.

But there may be complicating factors that users need to understand before buying. You may have to explain which other systems can interoperate with yours, or who owns the user's data and what they can do with it. (The technology and value of interoperability is discussed in further detail in Chapter 10.) You might have to guarantee how far into the future you will maintain the Internet service (if your company is acquired, goes bust, or discontinues the product).

The entrepreneur and academic Steve Blank describes four types of market in which a product can operate (see Figure 4-6).<sup>4</sup> The type of market influences how you position the value of your product.

---

<sup>4</sup> Steve Blank, *The Four Steps to the Epiphany: Successful Strategies for Products that Win* (K&S Ranch Press, 2005).



**FIGURE 4-6**

Four types of market in which a product can operate

The following sections outline what this might mean for IoT products.

### **A new product in a new market**

Embedded connectivity and intelligence will fuel the appearance of new classes of product and new markets. In consumer terms, the challenge is often to convince users of your vision. You have solved a problem they didn't realize they had, or had just accepted as "the way things were." The Glowcaps pill bottle top, mentioned in Chapter 1, reminds users to take their medication and helps the patient's doctor track how frequently it is taken.

### **A new type of product in an existing market**

Here, the challenge is to convince users that your product is the best solution to the problem. Perhaps it has better features or better performance. In IoT, these products may be familiar physical devices newly enhanced with sensing or connectivity (e.g., the Withings bathroom scales). Users need to understand the value that is added by the enhancements, such as easier weight tracking. They need to decide whether it's something they want, especially if it costs extra.

It might also be a technology that offers a step change in experience design. For example, airport terminals can be large and confusing. You would normally rely on signage to find your way around, but this isn't always clear, consistent, or guaranteed to tell you what you need when you need it. You don't want to miss your flight, but nor do you want to end up sitting around at the gate for too long because you were cautious and got there too early. Beacons (described in Chapter 2) offers precise indoor location. Several airlines have been trialing the use of beacons to provide passengers with in-context information and directions (see Figure 4-7). Passengers can be directed to the correct gate more easily, based on their current location in the airport. If a passenger is running

late but is very close to the gate, knowing his location might help the crew decide to wait. And if their plane is delayed, the app could provide them with a voucher to a nearby restaurant or café.



**FIGURE 4-7**

Illustration of an airport iBeacon trial (image: SITA)

### **A low-cost entrant to an existing market**

The falling cost of embedded computing enables cheaper alternatives to systems that used to be prohibitively expensive. For example, Lowes Iris (see Figure 4-8) and SmartThings offer DIY home automation kits at a far lower cost than professionally installed systems. You may be aiming the system at people who could not previously afford this category of device, or trying to convince those who could that you're offering a worthwhile saving. Either way, it's important to convince users that the system performs the basic functionality just as well as more expensive options. Any compromise needs to be something that doesn't matter too much. You need to be clear up front how you have achieved the cost saving: is the hardware cheaper? Does the system involve more work from the user (e.g., DIY setup)? Does it provide them with less personal (e.g., automated or lower bandwidth) customer service?

### **A niche entrant to an existing market**

Augmenting an existing product type with connectivity and potentially intelligence can create opportunities to address previously unmet user needs in an existing market. It may target a niche with specialist interests: for example, an energy monitoring system designed for those who



generate their own electricity and may sell it back to the grid. Or it may introduce a premium product for those willing to pay more. The Nest thermostat offered the first intelligent heating solution with high-end hardware and polished UX design in a market previously dominated by ugly, unusable plastic boxes. This reshaped consumer expectations of what a heating controller could be, even in the part of the market that couldn't or didn't want to pay extra for a Nest.



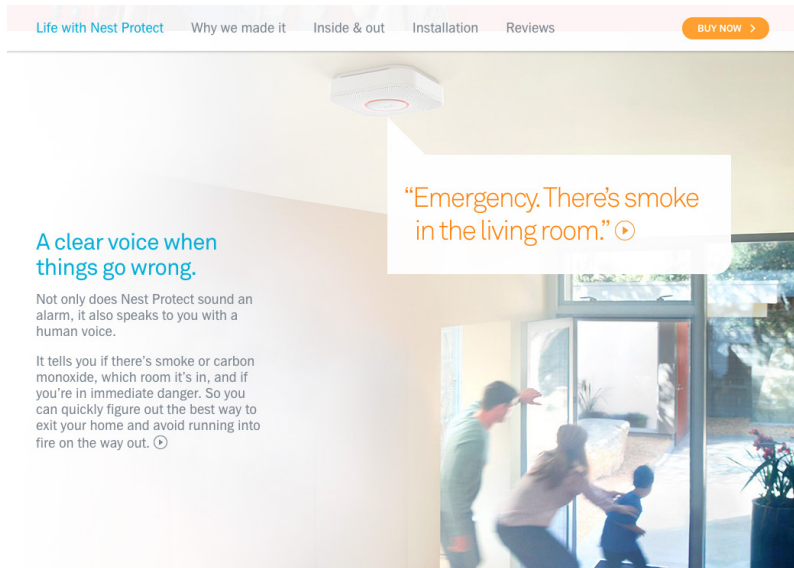
**FIGURE 4-8**

Lowes Iris Safe and Secure DIY home security kit—hub, motion sensor, two contact sensors, alarm keypad (image: Lowes)

## Tools Versus Products

For some specific connected devices, like a heating controller, there's a close mapping between function and value. It's easy for people to understand what it does. That's not enough to make it a *good* heating controller. But it's pretty clear what it does, and why you might want it. It will keep the house at a comfortable temperature, and perhaps save money. Devices that are enhanced versions of preexisting product types, like bathroom scales or baby monitors, have the advantage of being recognizable as things that meet a defined, familiar set of needs. You may have to convince customers as to why that product benefits from connectivity. And you may have to address concerns they have about adding connectivity or technical complexity to the product, such as security, privacy, or usability. But at least the product is familiar.

Mass-market consumers, in areas in which they do not have deep technical or domain knowledge, generally expect a product to come designed and engineered to fulfill a specific need. The Nest Protect smoke detector and carbon monoxide alarm is a good example of a *product*.<sup>5</sup> The marketing website focuses on the ways in which it is a better safety alarm (see Figure 4-9). Connectivity is only briefly mentioned, to say you'll be alerted on your phone if there's a problem when you're away from home.



**FIGURE 4-9**

Excerpt from the Nest Protect marketing website (image: Nest)

But many IoT services and devices can be configured to meet all kinds of needs. The onus is on the user to define their own needs and configure the device (or service) to achieve them. These are not products, but tools. Tools are often general-purpose devices, such as contact or motion sensors. The device has no inherent value to the user. The value comes when they are applied to solve a particular need, such as detecting intruders in the home, or warning you that you left a window open.

5 Nest Protect has suffered from some interaction design issues related to the ability to disable false alarms (see Chapter 8). But the Protect is still a good example of a clear product concept.

The Belkin WeMo Switch (see Figure 4-10) is a tool. It can be used to turn power to any appliance on and off remotely from a smartphone, or using an automated schedule. But it's up to users to define their own problem, realize that a smart plug could help, and configure it to solve the problem. An imaginative leap is required. In reality, many smart plugs end up being used on lamps. In our own research, users struggled to think of other uses for them (although ensuring hair straighteners/curling irons were turned off was a popular suggestion).



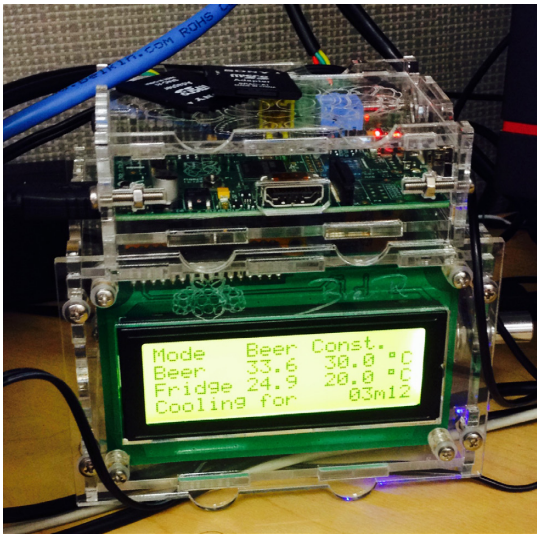
**FIGURE 4-10**

WeMo Switch and app

Services can be tools as well. The aforementioned If This Then That (which can also be used to control WeMo Switches), aims to make it easier for nontechnical users to link up and program devices and services.

Tools aren't bad. They can be very powerful for users with technical or domain knowledge. Users who have the time and motivation to configure a system to meet their own very specific needs and aren't daunted by the need to learn the system may really enjoy this process. This could be the home brewer who enjoys rigging his or her own fermentation chamber out of an old fridge (see Figure 4-11). Or a horticulturalist might be motivated to learn about the technology to configure

a remotely controlled plant monitoring, watering, and feeding system. Tools give us the possibility to be creative and take control of our environment.



**FIGURE 4-11**

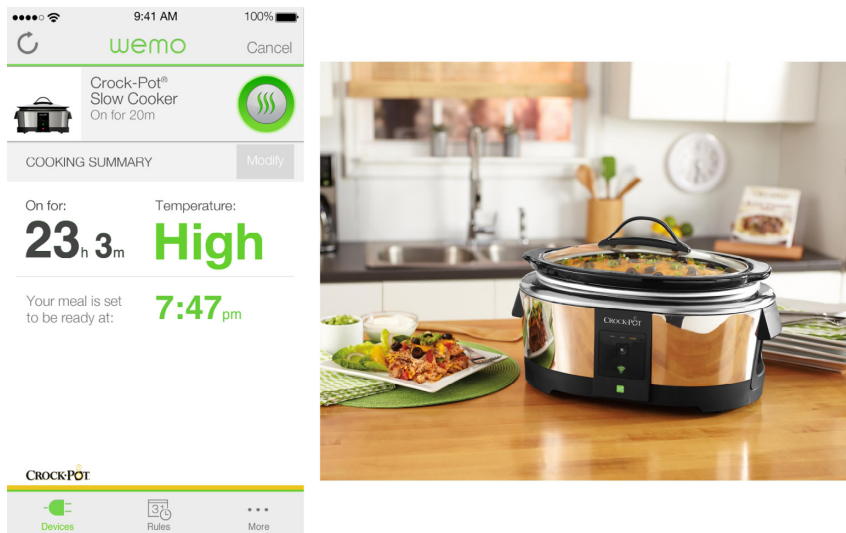
BrewPi is a fermentation temperature controller for brewing beer or wine; running on a Raspberry Pi computer and Arduino,<sup>6</sup> it comes with a kit to convert a standard home fridge or freezer into a fermentation chamber and is controllable via a web interface (image: Anthony Plunkett)

The IoT market, to date, has tended to create tools for innovators and early adopters. In an immature market that is exploring possibilities, that's fine. But it has tended to assume that the way to reach a mass audience is to make better-designed tools.

You can't turn a tool into a million-selling product just by making it usable. The WeMo Switch comes with a well-designed smartphone app that walks users through the setup process fairly clearly and makes it easy to set up rules to control the plug. But the onus is still on the user to use the plug creatively. It's not actually the *plug* they want to control: it's the appliance. Controllable plugs are simply a first step in the journey toward controllable appliances.

<sup>6</sup> At the time of writing, the Arduino model is being phased out for a newer version based on the Spark Core development board.

In spring 2014, Belkin and appliance manufacturer Crock-Pot released a controllable appliance: the Crock-Pot Slow Cooker with WeMo (see Figure 4-12). This allows the user to control the temperature and cooking time of a Crock-Pot remotely from a smartphone app. The context of use is a perfect fit for connectivity and remote control. Their value proposition is convenience: the meal that cooks itself while you're out all day. Remote control increases that convenience by allowing you to adjust the timing if you're home late. And being able to keep an eye on the device alleviates any anxiety about leaving a hot thing unattended in an empty house. It may be a niche appliance, but it's a well-formed product solution.



**FIGURE 4-12**

Crock-Pot Slow Cooker with WeMo and smartphone app (images: Belkin)

Mass-market consumers don't necessarily lack the knowledge, skill, or imagination to solve their own problems. They may be perfectly capable of doing so, but simply lack the time or have other priorities. At best, they might only have time to solve a few of them.

There is a rich market for products that solve their problems for them!

## What Makes a Good Product?

Good products seem to appeal to common sense, and new good products are often greeted with the reaction “I can’t believe no one thought of that before!” But developing good products can be far harder than our 20/20 hindsight might lead us to think. This section looks at the general qualities of a good consumer product before considering what features come with IoT.

### **THE PRODUCT SOLVES A REAL PROBLEM (AND MAKES THIS CLEAR)**

Most products are acquired in order to solve a problem for the user. A good definition of the problem, and the audience, is essential to creating a clear value proposition. This is the definition of what your product does for people, and why they might want it.

A clearly communicated value proposition is fundamental to user experience. When people come across a product (or service), they try to form a quick judgment about its purpose, and whom it is for. If it’s not immediately clear what the value proposition is, it may be dismissed: either because it is too hard to figure out, or because it does not appear to do anything of value for that person at that time. Worse, potential users may wrongly assume it is able to fulfill a purpose for which it is not really suited and waste time and/or money on a fruitless endeavor. (You may be happy to take their money in the short term, but over time too many unhappy customers will damage your reputation!)

It’s all too easy to end up with a poor or unclear value proposition despite good intentions. This is often the result of failing to identify the right problem for the right audience. You might have added features to show off what the system can do, or because they are simple to build, dictated too much by the capabilities of the technology at the expense of the original purpose and user needs. Or maybe there are competing interests involved in feature scoping. It’s common for systems to try to do lots of things. That may create a great tool for early adopters who like to tinker and customize, but it risks muddying the value proposition for a mass-market audience. Imagine you’re making a wrist-top device for outdoor pursuits like hiking or climbing. The core features are an altimeter, barometer, compass, and perhaps GPS. It might be quite straightforward software-wise to add on a calendar, to do list, and maybe even games. You can probably imagine a situation in which

someone, somewhere, might use those features. But you'll be at risk of obscuring the key purpose of the device: helping users find their way and stay safe. Too much flippant functionality might even undermine the perception that the device offers good quality in its core functionality. And it will make it harder for users to access the key features they most want and need.

If your device can fulfill multiple purposes for the user, you'll have to invest extra effort in helping users understand its value. A home contact sensor is a generic piece of hardware with no inherent value to the user. The value is in the function it enables: used to detect when an intruder has forced a door open, or when a medicine cabinet has been opened. Early adopters may love the flexibility to use the sensor as a tool that can do all kinds of things. But you'll have to help mass-market users understand what it could be for. For example, your app might offer specific window or cupboard alarm functionality to go with the device, even if these do much the same thing under the hood.

Connected products intended for the mass market need to demonstrate a clear advantage over any predecessors. Connected things are not inherently better than nonconnected things, just because they are connected. Despite being demo-ed at consumer electronics fairs year after year, the much-maligned Internet fridge concept has so far felt like a solution in search of a problem. Research shows that people can imagine using intelligent fridges that provide information about their contents, nutrition, and health, but this has not translated into demand.<sup>7</sup> Tasks such as managing shopping lists and looking up recipes simply don't feel as if they require a new, fridge-based screen. The idea of the fridge that automatically orders more shopping when goods run out is fraught with potential for irritating errors. If you have to make the fridge sync with your calendar or heating thermostat to see when you're on holiday in order to stop your regular milk order, maybe it's just simpler to buy your own milk after all.

---

7 Matthias Rothensee, "User Acceptance of the Intelligent Fridge: Empirical Results from a Simulation," *IOT '08 Proceedings of the 1st International Conference on the Internet of Things*, 123–139.

Connected sensors enable many kinds of data in the world to be captured, quantified, and made visible. Fitness tracking and energy monitoring (see Figure 4-13) are obvious consumer examples of this. But beware you're not just counting things. Data should be used to provide genuine insights that users can act on.



**FIGURE 4-13**

The Efergy energy monitoring service helps users understand their electricity consumption (image: Efergy)

For more information on designing with data, see Chapter 13.

*Connectivity* can enable remote control of devices. The core value of connected sockets and door locks is usually remote control (see Figure 4-14).

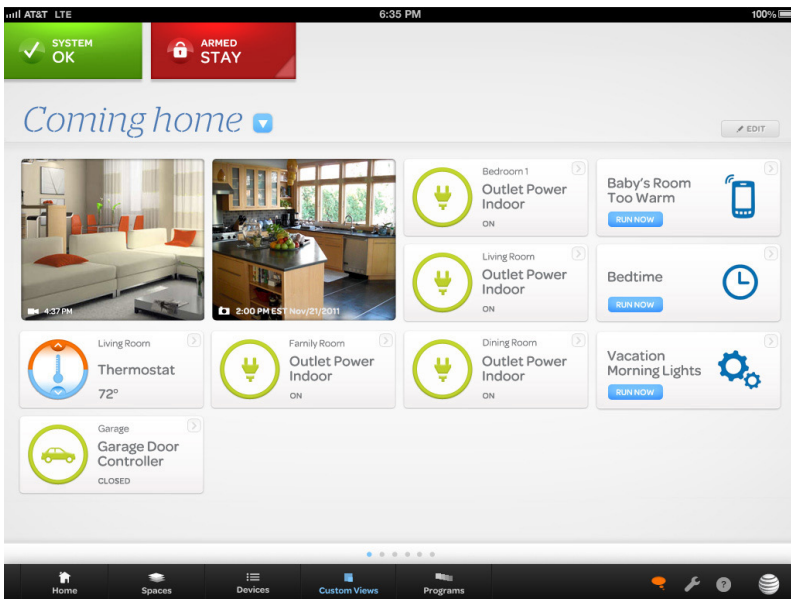
Connected home systems that allow automated rules to be created are examples of products whose main value is in automation (see Figure 4-15). Intelligent systems such as the Nest thermostat may promise to do the job (such as setting a heating schedule that best fits home occupancy) better than a human.





**FIGURE 4-14**

The August door lock, app, and hub (plugged into outlet; image: August)



**FIGURE 4-15**

An automated “coming home” smart rule in the AT&T Digital Life tablet app (image: AT&T)

Tags or sensors embedded in objects allow them to be trackable and identifiable. The FedEx SenseAware service (Figure 4-16) embeds a multisensor device inside sensitive shipments (such as medical supplies), allowing the sender to track the location of a parcel and the temperature, light levels, humidity, and atmospheric pressure to which it has been exposed. If any of these fall outside a set range, a replacement parcel can be dispatched.



**FIGURE 4-16**  
FedEx SenseAware sensor and web app (image: FedEx)

It goes almost without saying that your system needs to be reliable enough to fulfill its promise. Glitches and outages are inevitable in most systems and early adopters will forgive these more readily. But if there are contexts of use in which you cannot afford failure, the product must be 100% reliable. For example, emergency alarms for elderly or vulnerable people must always work. You'll need a backup power supply and connectivity (see Figure 4-17), and regular checks to ensure these work.



**FIGURE 4-17**

The hub of the Scout security system has a backup battery and 3G cellular chip so it won't stop running during power and Internet outages (image: Scout Security)

### **THE PRODUCT COMES AT A COST PROPORTIONAL TO THE PERCEIVED VALUE**

A good product needs to balance the cost and effort required from the user against the value it delivers. If the value is very high, users may be prepared to pay more, or invest more time in configuration.

Determining a price point is a tricky matter in itself. You'll have to consider manufacturing costs, competition and market conditions, and what users are prepared to pay.

You'll also need to consider the cost to the user of switching from whatever they were using previously. Household technology, like heating and alarm systems, tends to last years and users won't want to replace working boilers, sensors, or other kit at great expense without a significant benefit.<sup>8</sup> If you can support retrofit—new technology that can easily be integrated into old systems—without greatly increasing the cost of your product, you'll increase the potential market for the product.

---

8 C.F. the model of shearing layers, which describes buildings as a set of components that evolve and obsolesce over different timescales. "Services," like HVAC and plumbing, are expected to last 7–15 years. This concept originates from architect Frank Duffy and was developed by Stewart Brand in his book *How Buildings Learn: What Happens After They're Built* (1994, Viking Press).

In the context of UX, the perceived cognitive effort to use your product and the time it will take to get it set up and working affect who will buy it, and why. Be careful in your judgment here. In the thick of a project when you are excited about your idea, it's easy to overestimate how motivated users are to invest time in your product.

Smart homes are a typical example here. It's been possible to connect up lighting, heating, appliances, and entertainment systems for around 40 years, as we saw earlier. But you needed to be an enthusiast to set it up and program it (or wealthy enough to pay someone else to do that). A niche of users has taken great pride in their automated homes, but others have found them fraught with support issues, technology failures, and a poor fit with the needs of other guests and residents. Mass-market users often view home automation with suspicion: home is a very personal context, and one in which we are often loath to introduce novel technologies that might break our established routines. Most of us don't want to have to do a load of programming just so we can turn lights on and off. We manage that well enough already and it's an effort to switch unless the benefits are really evident.

Adding extra cognitive effort to everyday tasks is a common risk. The designer Scott Jenson proposes the idea of the “surprise package”: the mature consumer product that is “enhanced” technologically, turning it back into an early adopter product. As Jenson puts it:

Companies take product concepts that are now far into the laggard range of stability and established behavior, and they change the product significantly. ... The new product is effectively repositioned ‘back to the front’ of the curve, creating a high-tech product that can only be used or appreciated by the forgiving and accomplished early-adopter group of consumers. This is where much of the consumer backlash appears, as safely mature and benign products such as TVs, radios, thermostats, home phones and even cars are turned back into early adopter products, and then sold to an unsuspecting laggard audience.<sup>9</sup>

---

<sup>9</sup> Scott Jenson,, *The Simplicity Shift* (Cambridge, UK: Cambridge University Press, 2002). Available from <http://www.jensondesign.com/The-Simplicity-Shift.pdf>.

TV is a great example. TV used to be an instant-on experience. We may have had less choice of channels and no on-demand services, but you could be watching *something* within a second or two of turning it on. It can now take minutes. You may be faced with software updates for your set-top box and/or connected TV (perhaps for apps you don't even want but can't delete), then minutes of navigating around a program guide or on-demand service using a cheap remote control poorly equipped for the job.

If your product is replacing an existing consumer product or way of getting something done, pay attention to what was good about the old way of doing things. Try to preserve that and enhance the experience, rather than adding new complexity.

### **THE PRODUCT IS PLEASING TO USE**

The hard-headed cost/benefit analysis is important for any product, but the best products speak to us on an emotional level, too. This is partly about aesthetics, but it's not just about bolting pretty design on top of functionality. We form an integrated impression of the functionality and design of the product, and how well that fulfills our practical and emotional needs and fits (and perhaps communicates) our sense of who we are.<sup>10</sup> Figuring out the right experience is about design as well as product strategy, which is covered in more depth in Chapter 5.

## **Services in IoT**

In this section, we'll consider the role of services in creating a viable IoT product. Why are services important, how do users think about products that have device and service components, and what kind of services should you consider offering?

### **IOT PRODUCTS COMBINE DEVICES AND SERVICES**

At the start of the chapter, we set out that an IoT "product" is frequently a hybrid of physical device(s) and service provision. At the very least, Internet connectivity is a form of service, and there may be other service components that your product needs to work properly and ensure a good experience.

---

<sup>10</sup> Lionel Tiger's *The Pursuit of Pleasure* is an interesting viewpoint on the anthropology of what makes products appeal (Boston: Little, Brown, 1992).

When people buy a product, they expect to have the right to use it for as long as they like. When the product is dependent on an Internet service, there is a reasonable expectation that that service will continue to be available, for taking it away would render the product at worst useless and at best limited. After all, you would not expect your home heating or lighting to cease to function because the company that produced the original system had gone out of business, or no longer wished to support you. The service forms part of that experience. The relationship between the device and service can vary.

There is an inherent tension here between the old world of physical products, and the new world of Internet/web services. On the Web, new services appear and old ones are “sunsetted” on a regular basis. This is acclaimed as progress. But a physical product is likely to come with expectations that it will last for at least a few years. If the service stops working, the lifespan of the device is shortened, creating landfill (and unhappy users). Service providers have a responsibility to ensure that they are able to maintain and improve their Internet services, so that the product has a reasonable lifespan.

## **HOW USERS UNDERSTAND DEVICES AND SERVICES**

Connected heating controllers and door locks are examples of systems where the device is likely to be the focus: we might call them *service-enabled devices*. Users view the device as the most salient part of the system. For example, Nest users are likely to say “I have a Nest,” referring to the thermostat to represent the entire system. (It’s far less likely you’d hear someone saying “I use Nest,” “I have Nest,” or “I have a Nest system.”) Because the device is so central to the UX, users will have high expectations of its design and functionality. The service enables remote access and smarter functioning, but in the user’s mind it is a way to control the device (see Figure 4-18).

A security alarm is an example of a system where the *service* is the focus: we might call it a *device-enabled service*. The alarm service is what users care about. The sensors and other devices are generally low profile and most of the intelligence sits in the Internet service or gateway software. You could add or swap out devices without affecting the core functions of the service.



**FIGURE 4-18**

Nest advertising puts the device front and center (image: Jason Morgan)

Key factors that indicate that the service may be the focus of your user experience, not the device itself, may be that:

- Interactions are distributed across multiple devices, so no single device is the center of attention
- Most functionality lives in the cloud service or gateway software (perhaps because local devices don't have much computing power)
- Devices can be added, removed, or swapped without changing the core functioning of the system

As the UX expert Mike Kuniavsky describes it, the device is an *avatar* for the service.<sup>11</sup>

The Oyster travel card (see Figure 4-19) is a stored value contactless smart card used on London public transport. It can hold various types of tickets or a credit balance for travel on the underground, trains, buses, trams, and boat services. (“Stored value” means that the credit is notionally held on the card itself, rather than in a separate account, as with a debit card.)

---

<sup>11</sup> Mike Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design* (Burlington, MA: Morgan Kaufmann, 2010).



**FIGURE 4-19**  
The London Oyster card

Passengers add tickets or money to the card itself via online purchase, ticket machines at stations, or by setting up regular debits from their bank accounts. They swipe the card on a reader at the start and end of journeys to validate their tickets or deduct credit. The Oyster saves time and money processing ticket office transactions and reduces the number of paper tickets. To encourage use, fares are substantially cheaper than paper tickets.

The Oyster card itself is not much of a smart object. It's just a piece of plastic containing an RFID chip and a small amount of memory. The RFID chip passes a unique ID to a reader when a passenger swipes in. The memory holds information about the tickets or money stored on it, so the reader does not need to contact the back office service in real time every time the user swipes the card. This speeds up the rate at which passengers can pass through ticket barriers, which is vital during rush hour. Readers transmit transactions to the back office in batches.

The Oyster card is an icon of London life, but it is really just an avatar for the service. Without the card readers or the ability to top it up it wouldn't be much use to you. The Oyster service involves smooth coordination between many different channels, such as the Transport for London website, the ticket machines, ticket offices and shops that sell top ups, the readers themselves, and the back office systems (see “Service Ecosystems”).



Technically, the Oyster card itself is not even an essential part of the service: Oyster can also be used via NFC-enabled phones and bank-cards. In the future, the dedicated Oyster card might even disappear, but the service will remain. But services are intangible, and avatars can provide a concrete, tangible focus that helps us understand the service.

Right now, IoT systems are still pretty novel and not well understood, at least by consumers. It's easy to look to individual devices as a handle to understand the system, whether or not this is accurate. (We've heard smart meter users refer to the in-home counter-top display as the "smart meter," and the actual smart meter as the "computer under the stairs"; see Figure 4-20). You might need to play up the role of the devices in communicating what your system does (*presenting it as a service-enabled device*), just to help consumers understand it.



**FIGURE 4-20**

A British Gas dual fuel in-home display for use with a smart meter (image: British Gas)

Over time, as we all become more accustomed to the products around us having intelligence and connectivity, our ability to understand connected products as services without depending on physical manifestations may become more sophisticated. The idea of a heating system without a visible controller, or a door lock without a visible lock may seem strange right now, but in time, as long as they work, we might be more open to such things.

It will probably always be appropriate for some systems to have highly visible devices, and for some to focus more on service design (see Table 4-1). The key is to pitch your UX to best suit your product, and the needs of your users.

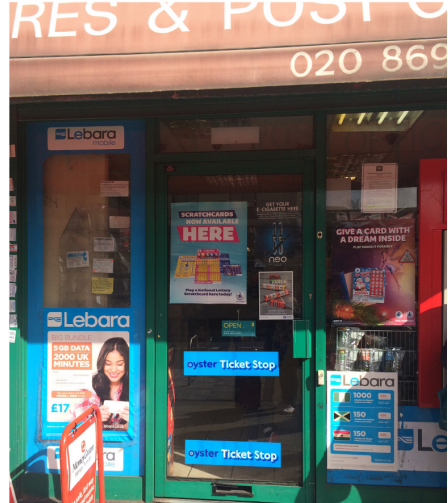
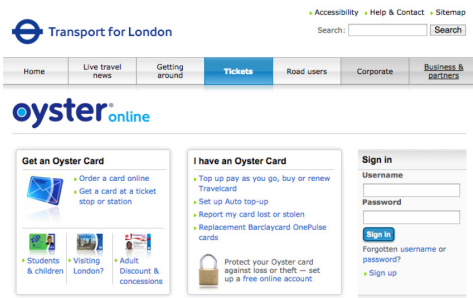
If your service is the focus of the UX, you can still make beautiful devices but make sure the service design is at least as good. And if the device is the hero of your UX, make sure it's attractive, usable, and does what it needs to do, elegantly.

**TABLE 4-1.** Service-enabled devices versus device-enabled services (many devices and services are somewhere in-between)

DEVICE IS THE FOCUS OF THE UX	SERVICE IS THE FOCUS OF THE UX
Industrial design of devices is striking	Industrial design of devices is low profile
Many interactions handled via the core device (e.g., a watch)	Interactions are distributed across multiple devices
Most functionality is handled on the core device itself (e.g., a washing machine)	Most functionality lives in the cloud service or gateway software
User views service as a way to control the device (e.g., "I have an app for my thermostat")	User views device as a means to enable the service (e.g., "The intruder alarm uses motion sensors")
Swapping the core device for a different make or model would radically change the user experience of the system (e.g., changing to a different brand of thermostat or make of car)	Devices can be added, removed, or swapped without changing the core functioning of the system (e.g., paying for travel with an NFC phone instead of an RFID card)

## SERVICE ECOSYSTEMS

Services are delivered through the interactions of networks of people, organizations, infrastructure, and physical components. The devices, and even the digital components, are only part of the experience. A part of the Oyster experience is the interactions you may have with station staff when buying a ticket or asking for help. In order to help you, they will have been trained to provide good customer service, but they will also need access to good information about the transactions on your card and system information. Making this whole system work smoothly is a lot more complicated than just making cards, machines, and a website: it requires someone to take a holistic view of how the service is experienced, and make sure all the components work reasonably smoothly together (see Figure 4-21).



**FIGURE 4-21**

The London Oyster ecosystem (website image: tfl.gov.uk)

Complex IoT services, such as a connected home, require the coordination of multiple devices working together, perhaps even using a degree of intelligence to automate some functions without explicit user instructions (e.g., turning off the electricity supply if a gas leak is detected). There may be multiple digital interfaces. There may even be coordination between multiple digital and physical services—for example, a heating system may use data from a third-party weather service, and the user might have the option of taking out a service and repair contract.

This is called an ecosystem.

If you're designing a service, you'll need to take a more complete view of all the parts of your service, and the relationships between them. For example, you may need to think about handling software and

component upgrades across your devices. Ovens may require new controller boards, and washing machines may demand firmware upgrades (see Figure 4-22). You'll need to design processes for handling these issues with consumers.



**FIGURE 4-22**

The Samsung WW9000 connected washing machine supports over-the-air firmware updates (image: Samsung)

There may well be customer support, marketing, sales, and perhaps professional installation and maintenance too. Mark Kawano, founder of Storehouse and former Apple User Experience Evangelist, describes how creating a well-designed product is about getting the entire experience right, not just about the UI layer we typically associate with design:

You see companies that have poached Apple designers, and they come up with sexy interfaces or something interesting, but it doesn't necessarily move the needle for their business or their product. That's because all the designer did was work on an interface piece, but to have a really well-designed product in the way Steve would say, this "holistic" thing, is everything. It's not just the interface piece. It's designing the right business model into it. Designing the right marketing and the copy, and the way to distribute it. All of those pieces are critical.<sup>12</sup>

<sup>12</sup> Mark Wilson, "4 Myths About Apple Design, From an Ex-Apple Designer," *Fast Company*, July/August 2014, <http://bit.ly/1bD2ezr>.

## BUILDING A SERVICE OFFERING

Thinking at the service experience level encourages us to take a broader look at user needs, not just the interactions with the website, mobile app, or embedded devices. Chapter 5 explores understanding user needs in more detail. In this chapter, we'll consider how this might create new opportunities to look at the wider service package you may offer customers.

For example, take home security. We typically think of alarm systems as something that makes a loud noise and acts as a visible deterrent on the home. A connected system can tell you when someone is breaking in, and perhaps film them. But if you're not able to get home, and no one else can respond, you can't do anything about it. Connectivity has alerted you to the problem but also enabled you to feel powerless to act.

In this case, you might have the option of paying for a professional monitoring service. The security firm (with your permission) can view your cameras before sending someone out.

IoT services often provide opportunities to capture data, which can be used to improve the service offered, perhaps through better customer service or smarter support. For example, diagnostic data about the functioning of a boiler could be used to identify systems at risk of breakdown and schedule an engineer visit *before* they fail. You might even package the cost of the service contract with the monthly fee for maintaining the service. Users may be happier to pay for something like this than to cover the cost of maintaining your Internet servers!

Professional installation or configuration may also be an opportunity for complex systems such as home automation technology. Time It Right,<sup>13</sup> designed for the needs of the Orthodox Jewish community, is a culturally specific example (Figure 4-23). There may in the future even be a role for independent IoT “plumbers” who specialize in helping consumers install, maintain, and repair connected home systems: a kind of role the technology blogger Anil Dash refers to as “blue collar coders.”<sup>14</sup>

---

13 <http://autotimeonline.com/>

14 <http://dashes.com/anil/2012/10/the-blue-collar-coder.html>



**FIGURE 4-23**

Time It Right comes with a professional installation and configuration service (image: Autotime)

Another service opportunity may involve helping users secure lower prices or better service from third parties, or otherwise benefit from the data that comes from your system. In the sustainable housing development of Little Kelham in Sheffield, northern England, residents have smart meters to track energy use and band together to bulk buy electricity (see the connected home case study between Chapters 6 and 7).

Personal services, such as installation or intensive customer support, are not necessary for all services and may not be practical within your business model. But it's worth considering the bigger context of user's expectations around the service you're offering, and how their needs will change over time, to make sure your service isn't missing something they think they need, or to spot opportunities to improve the overall experience.

## Business Models

Establishing the relationship between your system and the surrounding service is, in part, deciding on your business model. Put crudely, this can be summarized in two questions: what will people pay for? And what do you need for production to be sustainable?

## WHAT IS A BUSINESS MODEL?

A business model is the blueprint for how a business creates value for customers, and makes money. For example, a classic business model is the “bait and hook” one used by printer manufacturers. They charge a relatively low price for the initial hardware but make money on toner cartridges.

The model maps out how the business will make money, either from increasing revenue (selling more) or decreasing costs. Increasing revenue can be approached by:

- Generating new business from new customers
- Generating *more* business from existing customers

And even a not-for-profit organization needs a sustainable business model in order to survive.

## HOW DO BUSINESS MODELS AFFECT UX?

Business models shape the way users perceive the value of the service and the fairness of pricing. This can make the product proposition more or less appealing. Users will approach the product or service with a positive, trusting mindset, or a more skeptical or even negative one. This sets the tone for the rest of their interaction.

For example, the major energy companies in the UK have recently come under pressure for perceived unfair pricing practices. All are rolling out smart energy meters, which generate data that can be used to offer customers tips on saving energy and therefore money. But customers who feel that prices have not been set fairly treat this money saving advice with skepticism. This has a knock-on effect on the perceived UX of the energy saving service. Issues of trust need to be tackled up front in the design, perhaps through presenting pricing in more transparent ways.

## DEVICE AND SERVICE MODELS

In the traditional product business model, the provider charges once, up front, for hardware. This is how we are used to buying, for example, cars or household appliances. For a long time, this was also how we bought software.

In service models (of which many digital models are examples), the provider charges for ongoing service provision. Music subscription services such as Spotify or storage like Dropbox are examples of service business models. A nonconnected product can be supplied as a service; for example, renting a car through a traditional rental company is a service. The customer pays for use of a car, not for ownership of a particular car. Adding connectivity to objects increases the potential for service models.

The choice of business model is a balancing act between where the customer perceives the value to be and what they expect to pay for, and what it costs you to provide. IoT is new, which means that what the customer expects to pay for is not always a reflection of the costs you may incur!

For example, your connected heating controller requires an Internet service to provide you with remote control via your smartphone app. That costs money to maintain, especially when you consider that the lifespan of a heating controller might be 10 years or more. The provider might factor that cost into the price they charge you for the product up front, but that might make it more expensive compared to competitor products. Or they might choose to charge you a regular fee for ongoing service provision. But they might find that customers aren't used to paying an ongoing fee to keep their heating working! In that case, the company might try to add value to this service, by turning it into a platform that also supports other devices (like connected lighting or energy monitoring), or adding extra service components, like a maintenance contract.

As already discussed, the perceived value may rest in different places for different types of connected device/IoT system. Is it the overall service users think is most valuable, or one or more of the devices that deliver it?

IoT is immature and there's perhaps a tendency for users to focus more on devices, as these are novel (as we suggested earlier). Service providers may focus more on services, as that's where the potential for long-term customer engagement sits. You may think that you are offering a health management service, but if your users see beautiful bathroom scales, then persuading them to pay for an ongoing service may



be tough. The key for a business model is that customers feel they are paying a fair price (whether in terms of money or sharing their data) for the value they receive, and you are making the money you need.

## **BRINGING DIGITAL BUSINESS MODELS TO PHYSICAL PRODUCTS**

Building digital services around physical products enables suppliers to apply novel business models, more commonly found in the digital realm, to physical products.

Combining physical devices and services is likely to lead to some interesting, novel, and disruptive business models that challenge our pre-conceived ideas of what it is to own and use a product.

Business models we are accustomed to in the digital realm might make their way into the physical world. For example, we are accustomed to using websites that are free at point of use but make money from selling eyeballs to advertisers. It's not a huge stretch to imagine that physical devices might be given away in exchange for advertising or user data. The ubiquitous computing researcher Pertti Huuskonen jokes about the freemium fridge. Your supermarket might give you a free fridge with a screen that forces you to watch an advert every time you open it. Or the fridge might track the products you put in it and eat, and where you bought them, and share that data with the supermarket, and other advertisers. Users could buy their own fridges, essentially paying for ad-free experiences or privacy, or get free or cheaper appliances in exchange for their eyeballs and data. In the future, privacy may be a rich person's luxury.<sup>15</sup>

More positively, there are benefits to digital service business models. There are opportunities to develop ongoing relationships with customers, understand more about the people who buy your products and what they do with them, and tailor services better to their needs. Users are accustomed to (and mostly comfortable with) web-based services that capture and store information about them to provide a better service. The sensing and processing capabilities of IoT devices open up potential to extend these personalized services to the physical world.

---

<sup>15</sup> Personal communication with Pertti Huuskonen; Pertti Huuskonen, "Run to the Hills! Ubiquitous Computing Meltdown," *Proceedings of the 2007 Conference on Advances in Ambient Intelligence*, 157–172, <http://www.cs.swan.ac.uk/~csmx/csrsRG.pdf>.

You may be able to capture user behavior that wasn't previously visible, perhaps in real time—for example, how people are using energy via smart metering, identifying and tracking people in a physical space, or monitoring traffic levels via aggregated data about the density and speed of movement of drivers' smartphones. Knowing how often or intensively a product is used, and what for, enables you to tailor the service, sell supplies, improve the next version, or offer additional services. For example, some Nespresso coffee machines now come with a SIM card, allowing Nespresso automatically to send more capsules when the customer is running low (see Figure 4-24).



**FIGURE 4-24**

The Nespresso Zenius coffee machine comes with a SIM card (image: Nestle Nespresso)

You could charge users based on their behavior. For example, some car insurance companies, such as Insure the Box in the UK,<sup>16</sup> base pricing on actual driver behavior rather than demographics. This may benefit responsible drivers who are in a demographic category considered to be at high risk of accidents, such as under 25s.

---

<sup>16</sup> <https://www.insurethebox.com>

Products can proactively maintain themselves or provide data to enable smarter support: as discussed earlier, boilers could identify when they are developing a fault and be serviced before they break down; appliances could share fault data with the manufacturer so customer support can help users diagnose and fix more of their own problems. It is possible to vary pricing based on time of use (e.g., charging more for electricity at certain times of day to manage demand).

A good UX design needs to balance the needs of the business with the needs of the user. Even if you are not shaping the business model of the product you are working on, you at least need to understand it. The best business models serve the interests of customers as well as the business. In the car insurance example, cheaper insurance is granted to drivers who would otherwise be penalized on grounds of age, but the model also allows the company to reduce costs and uncertainty through more accurate risk profiling. User and business needs can be in tension: for example, a connected home service might rely on heavy upselling of new products. Here, design can make the difference between the upsell advertising being either useful or at least minimally intrusive, or downright irritating or something that causes users to stop using the system altogether.

## Summary

A clear *value proposition* ensures users understand what your product does, and whether they want it. This is essential in order for them to understand how it works, and how to use it.

Innovators and early adopters are inherently interested in technology and forgiving of imperfections. Mass-market consumers often have different needs.

Many IoT systems are *tools*: they require the user to frame their own problem and configure the system to solve it. Consumers tend to look for *products* that promise to solve a particular problem for them and come already configured to do that. They expect the cost and effort of using the product to be in proportion to the value it brings them.

IoT creates new opportunities for information gathering, sharing, remote control, and automation. But there are common pitfalls that can limit a product to early adopter markets. In particular, be careful of introducing *new* complexity to mature consumer products.

The UX of an IoT product might be focused around the device or the service. All IoT systems depend on some kind of digital service, and perhaps offline service components too, like professional installation, maintenance, or customer support helplines. Ensuring these work well together is an important part of the overall UX.

*Business models* shape the way users perceive the value of the service and thus the UX. Bringing connectivity and intelligence into devices may lead to digital business models appearing in the physical world.

# Understanding People and Context

**BY ELIZABETH GOODMAN**

Most people do not have limitless funds and patience for untried products and services. They tend to have habits that they don't want to change. They have limited budgets. They are not enthusiastic about learning new technical skills. Moreover, they have an inconvenient tendency to ask themselves whether a new purchase is really worth the effort, expense, and disruption to their routines. The answer to this question, as many makers of connected products have found, is usually “no.”

In this chapter, we'll show how early engagement with potential customers and users can help you set up connected products and services for mass-market success and avoid expensive and wasteful failures.

This chapter introduces:

- Concepts and questions to guide early investigation of potential audiences and contexts of use (page 153)
- Methods and tools for answering those questions (page 169)

This chapter addresses the following issues:

- How to define product roles beyond that of “the user” (page 157)
- Why it's important to identify where needs and roles conflict (page 161)
- What contextual factors matter most to connected product design (page 162)

Throughout this chapter, we'll be using a hypothetical home security design project to illustrate key points.

## [ NOTE ]

This chapter focuses on questions and methods specific to connected products and services. If you're not already familiar with more general product design research methods, try supplementing this chapter with a short introduction to the field, such as *Just Enough Research*, by Erika Hall (<http://abookapart.com/products/just-enough-research>).

## The Role of Research in Connected Product Design

Consider the sad career of the “smart fridge” (Figure 5-1).



**FIGURE 5-1**

The Electrolux Screenfridge, 1999 (image: Electrolux)

Despite massive publicity attending each launch, no “smart fridge,” from the 1999 Electrolux Screenfridge to the 2012 LG Smart ThinQ, has ever attained market success. By 2014, the SmartThinQ was discontinued. Yet every few years, a new smart fridge has appeared. It’s amazing how many obviously intelligent and talented people keep sinking resources into an idea whose time just never seems to come.

There's no way to guarantee success. But the best way to avoid making the next smart fridge is to remember that *you are no different from those intelligent and talented people*. The best way to avoid repeating the past is to ask some hard questions—starting *right now*—about what you're going to make, who would actually use it, and what support it needs to work. That's the proper role of UX research. In this chapter, we'll introduce early stage exploration with potential customers and users. Later, in Chapter 14, we'll introduce rapid product prototyping methods to speed up iterative cycles of design, evaluation, and redesign.

## Initial Questions and Concepts

Imagine you're about to leave for a long vacation far away from your home. You're a little worried about fire, burglars, or even just missing packages. However, none of the home security systems on the market seem right for you. You think: there's *definitely* some design opportunities here.

But what kinds of opportunities? Do you want to be able to unlock doors remotely? Turn lights on and off to mimic normal occupancy? Have the house automatically alert the police in an emergency? Maybe what you want isn't a home security system at all, but something that works more like traditional apartment building concierge, which can automatically alert a local contact if you're away.

The options proliferate as you dream up a new home security product to suit your needs. And each product connects a different set of people, places, and things. Let's list a few elements of a generic home security service:

- The inhabitants of the house
- The police
- Any humans contracted to be on call
- Software designers and developers, whether in house, contracted, or third party
- The manufacturers of any hardware and casing (likely different factories, perhaps even on different continents)
- Relevant government regulators, such as the Federal Communications Commission in the United States

- Telecommunications networks and their associated protocols and constraints
- Product technical support staff
- Industry journalists

...and the list just keeps going. Users are important, but product and service design requires cooperation from an entire web of people, organizations, and technical systems. Just think about one of the earliest massively popular connected appliances—the home radio. All it could do was select the station and change the volume. Success depended on the excitement generated by radio station broadcasts.

To design connected products as successful as the first radios, we need to start with the people who will use them. But we also need to think about the web of relationships those products need to thrive, just as radio sets needed stations and broadcasts.

## **ACTORS AND STAKEHOLDERS**

Think about the home security elements we just listed. Very few of the people on it are actually users. Some of them, such as the police, might never know the product exists. As products and services get more complicated, it's helpful to get more specific about the kinds of people in our web. We'll talk about two here. *Actors* directly handle the product. They physically touch or see it. *Stakeholders* are organizations or individuals who don't directly interact with the product but who still have an interest in how it works.

### **But first, how not to define people**

In product marketing, it's common to use demographic attributes (age, sex, income, ethnicity, religion, etc.) to segment markets into different audiences. In UX design, actors should not only, or even mostly, be defined by demographic attributes such as age, income, or ethnicity. In fact, driving UX with demographics is generally a bad idea.

Consider this real-world example: a group of device designers at a major technology company were getting angry complaints about the error-filled data entered by long-haul truck drivers into mobile terminals. The team at first assumed that the truck drivers were uneducated and careless, and that errors were unavoidable. Yet the errors got worse and complaints mounted. Finally, in the dead of winter, a senior designer decided to hit the highways to see the problem himself.



The drivers, it turns out, could spell just fine. The problem lay in the device's tiny buttons. The drivers were often big men with correspondingly big hands. Especially in winter, while wearing bulky gloves, hitting the wrong keys was inevitable. Unfortunately, the device provided no easy way to fix mistakes before submitting data. The solution was simple: redesign the interface to require less manual typing, and add a big OK button to aid error correction. Error rates plummeted after the introduction of the redesigned device.

Unconsciously stereotyping is easy to do when you rely on demographic statistics. It's true that most of the truckers did not have college degrees. But that didn't make them lazy. Or unable to spell. The problem was, ironically, related to another demographically related attribute: the median size of trucker fingertips. The designers could have lowered the error rate months earlier (and avoided irritating both customers and the truckers!) if they had done a little in-person research early on.

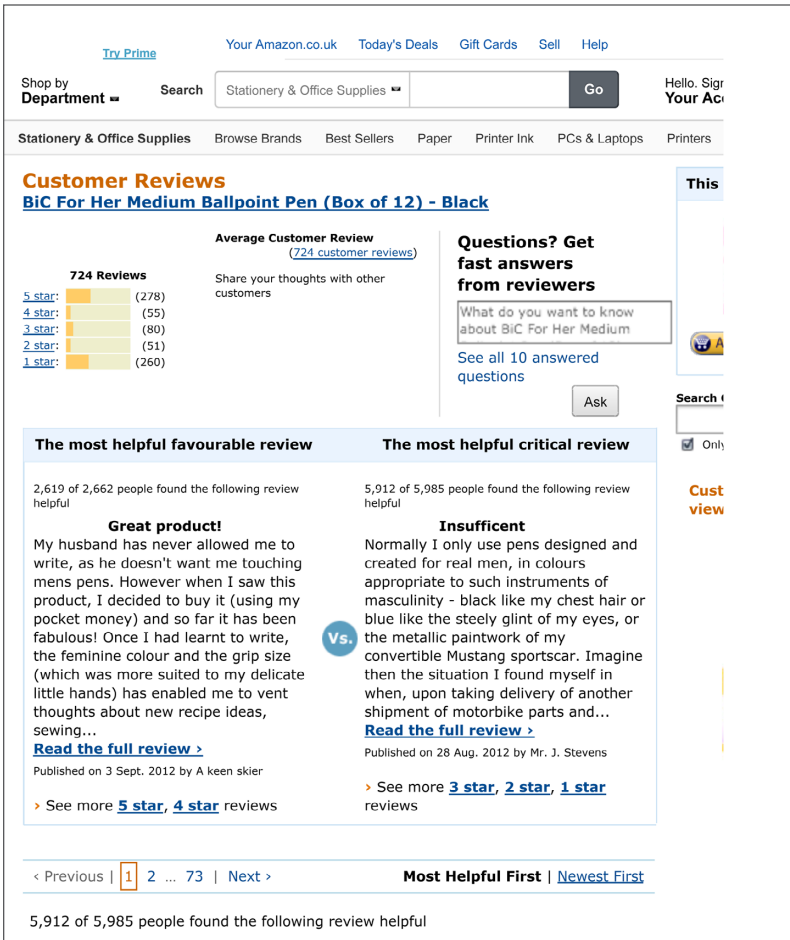
When you stereotype based on demographics, the best-case scenario yields a product that's less useful than it could be, as with the truckers. Worst case, you end up alienating your intended audience, as with the BiC for Women Amazon disaster (Figure 5-2).

Are some relevant connected product design factors linked to demographics? Sure, including:

- Body dimensions and product ergonomics
- Availability of resources like time, money, or space—for example, people raising kids are likely to have less solo leisure time... although you can't be sure!
- Diminished or increased physiological capacities, such as vision, hearing, agility, strength, or memory<sup>1</sup>
- Preferred language and other localization requirements

---

1 However, lessening capacities may lead to compensation elsewhere. People with diminished vision often astonish fully sighted people with the sensitivity of their hearing. So, again, it's wise to check one's assumptions.



**FIGURE 5-2**

BiC releases BiC for Her pens; hilarity ensues<sup>2</sup>

However, questions about activities should drive product strategy. What do actors need to accomplish their goals? Or even: what are their goals in the first place?

When starting to study users, you may need to make some decisions based on demographic attributes, but you should focus your effort on people who likely share the same goals and needs. Demographic

<sup>2</sup> Amazon.co.uk: Customer Reviews: BiC For Her Medium Ballpoint Pen (Box of 12) - Black. Retrieved from <http://www.amazon.co.uk/product-reviews/Boo4FTGJUW>.

attributes, then, are generally helpful only insofar as they affect what users need a product to *do* for them—such as automate heavy lifting or give them more time to spend with their kids. The trick is to treat demographics as a *source of questions*, rather than a shortcut to answering them. Try asking yourself:

- What do you *think* you know, based on demographics?
- What evidence do you have for those suppositions?

If your evidence is, “Well, everyone knows *that*,” or “That’s just the way those people *are*,” then what you have is an assumption—one that you should check with research.

Without early engagement with representatives of that group, “the user” becomes a dangerously vague character. Empathizing with “the user” is unhelpful if those needs, capacities, and emotions don’t actually exist—as in the story of the truck drivers. Basing team decisions on “the user” will not help if team members all have different ideas of who “the user” is and what “the user” wants.

You can change your mind about the intended audience for your product midway through research or design—that’s what iterative design is all about. But if you never consider the question in the first place, you’ll be guessing at what to prototype. You might also miss some market-revolutionizing functionality that a little bit of targeted investigation could inspire.

### **Actors, roles, and expertise**

An *actor* can be an individual (e.g., the person using it) or an organization (e.g., a corporation selling it). To start investigating actors, map out a potential product lifecycle from design to disposal. Let’s walk through a few important actors in the life of a home security system.

There are the makers of the product: designers, engineers, and so on. Makers can also include service providers, like marketers. Makers can include vendor organizations: perhaps one or more hardware manufacturers, a software platform provider, the installation and maintenance engineers, and consumer-facing sales, support, and so on.

There are, of course, users. But even a relatively simple home security system can have a variety of user roles. Imagine a family with a teenager, a young child, and a pet cat. The parents are *buyers*. They choose the system and pay for it. The children only need to turn it on

and off. The cat, meanwhile, is also a kind of user. If it isn't taken into account, the alarm might go off every time it jumps through a window. What about delivery people or short-term guests? They shouldn't have much control over the system, but do need occasional, perhaps revocable, access.

Then imagine how that same security system might work for an older person living alone, with a worried family living far away versus a set of mostly friendly but largely unconcerned housemates. If the security system is purchased for a furnished rental or collective housing arrangement, the buyer might be a stranger to the users, and vice versa. A family might have different expectations for a system installed by the parents than for a system provided by an anonymous real estate management firm.

From installation to repair, connected products will require technical support. When the security system needs repair, we encounter the *technician*. That technical support may come from a professional—or perhaps it is the teenager or a parent who manages the Internet provider and changes the batteries. Who will install, troubleshoot, and repair your product? What do those technicians need in order to do their jobs well?

Then, there are the actors involved in disposal. Perhaps it's sold to another user. Perhaps it goes to an ewaste recycling facility. Perhaps plant workers have to incinerate it. Disposal is a predictable stage of a product's life—those who take part in it safely (or unsafely) are also actors.

How *expert* do you expect actors to be? What kind of knowledge or skill do they have already? What do they need to learn? In rented housing with little stability, the security system shouldn't assume transient inhabitants stay long enough to master complex functionality. A system designed for a commercial setting like a small daycare, however, could reasonably assume some required employee training.

Technical “expertise” comes in many flavors. Even if you can assume users and technicians are very familiar with security systems, they may not be so good at configuring network settings, and vice versa. So the home security product you're designing would need to accommodate varying levels of expertise with systems for controlling access and

remote monitoring. That’s where research comes in. Once you have decided what kinds of actors might be involved, you can begin to figure out what they know—and *don’t* know.

For users who are also buyers, the purchase process is part of the user experience. When users are not also buyers, decisions taken at the time of purchase shape what people use, and how they expect to use it. Later, user experience depends on users’—or technicians’—expertise in diagnosing and repairing inevitable technical problems. Table 5-1 lists some common questions about actors to answer with research.

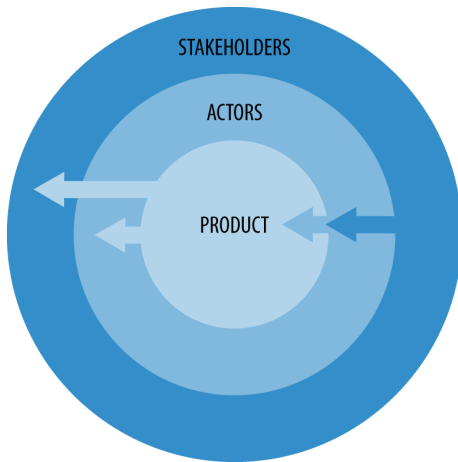
**TABLE 5-1.** Starter questions about actors and roles

ROLE	RESPONSIBILITY	COMMON QUESTIONS
Customer/buyer	Makes the decision to purchase the product	Why should I buy this product? What makes it better than competitors? Is it worth the cost?
User	Triggers everyday interactions with the product	Why do I want this product? What do I want from it? How do I use it? Is it easy for someone like me to use? Will using it change how other people think of me?
Technician	Installs, maintains, and repairs the product	How do I diagnose the problem? What resources (e.g., information, tools, skills) do I need to fix it?

### Stakeholders

Unlike actors, stakeholders don’t directly interact with the product itself. Stakeholders can be individuals or organizations that indirectly affect how a product works, and those who are indirectly affected by its use. Figure 5-3 illustrates the relationship between actors and stakeholders.

Some stakeholders are people. Local police officers might be stakeholders in a home security system. Police officers do not buy, use, or maintain the home security system. However, how quickly and politely they respond to alarms will affect how users feel about the product. People affected by the product *even though they don’t use it* are also stakeholders. Neighbors irritated by constant alarms and police visits, for example, could quickly make life difficult for the users of our potential home security service.



**FIGURE 5-3**  
Stakeholders and actors

Stakeholders can also be organizations. Later in the chapter, we'll discuss laws and regulations as part of the product context. Government regulators are important stakeholders for any product that falls under their mandate—which is basically any product or service that has a nondigital component. Car service innovator Uber, for example, notoriously fell afoul of insurance regulators<sup>3</sup> and taxi unions. As the company grew, insurance companies no longer wanted to allow drivers ferrying passengers around all day to pay premiums based on only occasional car-sharing. And let's not forget the angry taxi drivers facing the loss of fares!

One of the key reasons to identify stakeholders *as indirect actors* is to make sure you keep their interests in mind. When influence is indirect, it's easy to overlook. But if how a product works in practice depends on the activities of the police or a government regulator, the success of your product may depend on early engagement with their representatives. There's no sense in bringing predictable and avoidable public anger or legal prosecution down on yourself. Interviews with subject matter experts, such as lawyers and local political stakeholders, can help you identify potential obstacles and how to work around them.

---

<sup>3</sup> D. Jergler, "Uber, Lyft, Sidecar Toe-to-Toe with Insurers State-by-State." *Insurance Journal*, June 27, 2014, <http://bit.ly/1bD4YNI>.

## When needs conflict

After identifying actors and stakeholders, we can begin to ask what our product should do for them. As always, the answer cannot be “everything” if we hope to ship a desirable, usable, well-priced product.

Ideally, you’d start by matching the needs of intended users to your own business interests and goals. For the purposes of product and service design, a *need* is anything we value highly. A need can be a tangible deficiency: *I need to find the nearest gas station*. Or it can be a more intangible aspiration or goal: *I need to feel like my friends care about me*. Sometimes “needs” seem frivolous to outsiders; sometimes they are literally matters of life or death. What matters for product definition, however, is that those needs are deeply felt and that you are interested in addressing them.

But what happens when deeply felt needs conflict?

Take Lively, a service that helps families monitor the physical condition of elder relatives through a connected wristband and set of sensors, and gives the relatives updates on family activities. Here’s how Lively ([mylively.com](http://mylively.com)) makes the case for itself:

You need peace of mind—but sometimes your family and friends *need it even more*.

“Peace of mind” sounds like a compelling value proposition for both the elderly people being monitored and the families who care about them. However, Lively’s value proposition also illustrates a particular challenge for connected products and services: conflicts between what different people in different roles want for themselves. Lively has to mediate a compromise between the desire of family members to ensure that a physically or cognitively impaired loved one is well...and the likely distaste of those loved ones for what may feel like spying or surveillance. If there is a fight over monitoring, who wins? The loved one? The family? Or will the family simply choose to skip the fight altogether by rejecting the product?

Lively’s challenge is not unique. In fact, it’s common. Many connected products live closely to their users. They are embedded into buildings, furniture, and cars, worn like jewellery, or even implanted into our bodies. Such products are almost guaranteed to raise sensitive questions of privacy, autonomy, and control. Sensors watch us; algorithms draw conclusions about our activities; actuators respond. We hope that the products that live with us are acting in our best interests...but whose goals are they really furthering?

In defining connected products, one of the most important functions of design research is to surface conflicting needs that might sabotage a seemingly well-conceived product or service. Not all conflicts can—or should be—resolved. It may be that one group will have to remain unsatisfied. The neighbors might just have to live with the alarms and police calls. However, the best chance for negotiating those sensitive questions is to identify where and how a product is likely to raise them in the first place.

## THE CONTEXT OF INTERACTION

By “context,” we mean the factors *outside the product* relevant to our interactions with it—the immediate physical environment, human intentions and actions, organizational policies and legal rules, and so on. Context always matters to design, of course. But the more that products sense and respond to their surroundings, the more variable the locations and times of use, and the more dependent products are on interconnected networks—the more crucial it becomes to investigate context.

In the following section, we’ll briefly introduce four different ways to look at context: operational, behavioral, ecological, and sociocultural. Figure 5-7 illustrates how those contextual factors might play out in a home security system in which a phone application monitors a suite of devices.

### Operational context

Operational context refers to the physical world outside a device plus the resources it needs to function. Most obviously, operational context includes *environmental conditions* (heat/cold, wet/dust). It can also



include *spatial arrangements*, such as standard architectural dimensions or clothing sizes. Successful installation of a home security system, for example, often depends on whether the user's doors and windows match what the maker expected.<sup>4</sup> What about *infrastructure*? Products intended for wilderness, rural, or unstable regions can't depend on reliable or centrally maintained Internet access and mains power. Under those circumstances, a truly reliable home security system would need to provide its own power and electricity.

*Payment and supply chains* might also be part of operational context. If your audience can't or won't pay for power and network access, then those resources may be practically unavailable even if technically present. Especially in remote areas, you may need to consider the likely logistics of repair and upgrade. It's no favor to sell a product that can't be fixed or easily replaced.

Research on operational context can inform business and service models for payment, distribution, and repair. It can also guide engineering requirements, particularly in coping with extreme environments, unreliable power, and intermittent network access. That's what motivates the "go anywhere, do anything" router start-up BRCK to leave their Nairobi office for what they call "expeditions" across the continent. The expeditions test whether the router can survive extreme heat, wet, dust, and vibration (Figure 5-4). As BRCK's Philip Walton writes:

It is simply impossible to know what Africa can throw at you until you get out there and experience it for yourself. We are learning a lot about what is required to actually be rugged-enough for Africa.<sup>5</sup>

But the expeditions also test the process of getting the BRCK online across countries and telecom providers (Figure 5-5) so that the team can streamline the process for users.

---

4 For a short but compelling run-through of the complexities of operational context, see: D. Raffel, "It's Often (a Lot) Harder Than It Looks," January 8, 2015, <http://bit.ly/1bD5jzq>.

5 P. Walton, "BRCK Expedition—A Dash South—Day 2," BRCK, November 15, 2014, <http://bit.ly/1bD5B9w>.



**FIGURE 5-4**

Checking antenna-enabled BRCK connectivity in rural Kenya using an Android monitoring app (image: Erik Hersman)



**FIGURE 5-5**

Documenting the process of buying a SIM card and data plan in Zambia (image: Mark Kamau)

### **Behavioral context**

By “behavioral context,” we mean: *time* and *place*, the *proximity* or *visibility* of other people, animals, and things to the main participants, important *physiological states of being*... or any other tangible, sense-able dimensions of a given event or activity:

### *Time and place*

What sorts of time-based rhythms might such a service need to learn and anticipate? A home security system might start by learning workday and weekend household routines. Over years of use, it could also adapt to predictable cycles of holiday plans and visits from family members. What events signal the arrival or departure of an inhabitant or guest?

### *Proximity and visibility*

Who is allowed to access the home? When and how? Moreover, any kind of security service will have to consider privacy. What kinds of activities are usually visible to watchers? Which are typically private?

### *States of being*

What kinds of vision, hearing, or other physiological capacities should we expect? For example, is it reasonable to expect users to remember numerical passcodes?

Identifying appropriate behavioral factors is often the first step in algorithmically modeling activities and responding appropriately. The difficulty lies in deciding which behavioral factors might accompany those activities, experimenting with how to sense those factors, and then figuring out how to translate sensor data into usable information. It's not a small job. In Chapter 6, we'll introduce ways to simulate sensing and responding to behavioral data in order to prototype devices such as fitness trackers or smart home gadgets.

**Sociocultural.** What expectations do people have for their own behavior? For other people? Who or what enforces them? And what happens when those expectations are violated? Sociocultural questions ask us to consider the reasons why we do some things...and not others. They ask us to consider how groups of people find similar meanings in objects and phenomena, and how those shared meanings might vary across different groups of people.

Research transforms the abstract and fuzzy concept of *values* into concrete, specific terms. What does safety or security in the home mean concretely to your audience? What specifically makes them *feel* safe and secure? How do they know they are *not*? Answering questions like these is the first step in articulating why your product idea is important (see Chapter 6 for more).

Business rules and governmental regulations may affect what kinds of services you can offer to users. What are the relevant *laws* and *regulations*, exactly? Earlier we mentioned Uber as an example of a profitable company dogged by regulatory concerns. What happens if your product stretches or breaks the law, and who judges compliance?

Most importantly, research into sociocultural context also helps anticipate controversy and discomfort. Take the case of Google Glass. It's perfectly legal (in the United States) to record video in public places without first getting permission. Taking photographs of friends is socially acceptable—even encouraged. Yet head-mounting a small video camera, as Google Glass did, provoked controversy. Bars banned it (see Figure 5-6), and wearers encountered visible disapproval. Journalist Matt Honan, who wore Glass for a year, writes:

Even in less intimate situations, Glass is socially awkward. Again and again, I made people very uncomfortable. That made me very uncomfortable.<sup>6</sup>



**FIGURE 5-6**

Sign banning Google Glass (image: LoKan Sardari)

Experiences like Honan's are why it can help to investigate sociocultural context by enacting scenarios and making prototypes—though hopefully less expensive prototypes than Glass (see Chapter 14 for

---

<sup>6</sup> M. Honan, "I, Glasshole: My Year with Google Glass," December 30, 2013, WIRED, <http://www.wired.com/2013/12/glasshole/>.

more). Sociocultural expectations are so taken for granted that often, the best way to see them is when expectations of correct behavior are breached.<sup>7</sup>

**Ecological.** Ecological context, as the name might suggest, concerns what's often called a *product ecosystem*: the networks of economic relationships that position an organization and its products in a larger world.

Designer Mike Kruzeniski, one of the creators of Microsoft's groundbreaking Windows 7 design language, writes:

A good phone isn't just the quality of the device in your hand, it's the sum of the apps and services and accessories and cases and connected devices around it. If we wanted Windows Phone to be a great product, we would need to think of the product ecosystem as being a design problem as important as the product software.<sup>8</sup>

It's hard enough simply designing a product in itself. So it's easy to forget about cultivating what Kruzeniski calls the "the spaces adjacent to the product": actors such as user groups and developer communities, and content contributors and distributors. Kruzeniski continues,

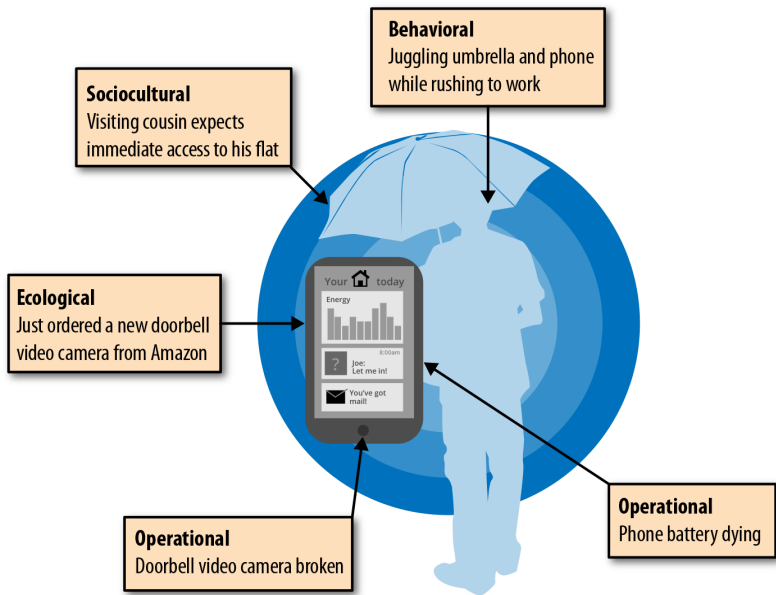
We needed to understand who would and wouldn't be interested in developing apps, why developers would or wouldn't want to build apps, what information was and wasn't helpful for developers to design apps, and what kinds of tools developers were going to need to get started.

That's why we do research into ecological context: to make sure that third parties will help make the product successful from the start. Thinking ecologically, you might ask yourself, how are potential users likely to hear about my home security service? If the original owner moves on, does the product/service move too, or does it remain with the house? See Figure 5-7.

---

7 This classic scholarly article on prototyping new products is worth reading: A. Crabtree, "Design in the Absence of Practice: Breaching Experiments," *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (2004): 59–68.

8 M. Kruzeniski, "Macrointeractions," <http://bit.ly/1bD6jUp>.



**FIGURE 5-7**

Sample contextual factors in a home security service scenario in which the primary user is trying to open the front door remotely for a house guest

Research into ecological context can also inform graceful handling of product “end of life” activities such as reselling or recycling—the kind of activities important to responsible device design (see Chapter 15).

In the next section, we’ll discuss some ways to answer the topics we presented and questions we posed (Table 5-2).

**TABLE 5-2.** Starter questions for each aspect of context

CONTEXT	OBJECT OF STUDY	QUESTIONS
Operational	The physical settings of use	What environmental conditions might the product encounter? What sorts of resources will likely be available to keep the product functioning?
Behavioral	Human activity and states of being	What kinds of activities are important to you? What tangible attributes characterize those activities?

Ecological	Networks of organizational, legal, and economic relationships	<p>What actors and stakeholders are involved in creating and maintaining a product or service?</p> <p>What do those actors and stakeholders need to participate and cooperate?</p>
Sociocultural	Shared ways of interpreting the world	<p>What expectations might actors and stakeholders have for themselves, others, and the product you are making?</p> <p>What happens when those expectations are violated?</p> <p>Who or what produces or enforces “appropriate” behavior for users, other actors, devices, etc.?</p>

## Techniques: From Asking to Watching to Making

So how do you go about investigating people in context? This section will give you some techniques and tactics that we’ve found work especially well for questions specific to connected products. These techniques are great to use *before* you commit significant time and money to any single design direction.

Documenting how your audience deals with the situation or activity that interests you is always a good first step. Here, we’re going to focus on activities that will help you understand the context of use and what it means to your users, including spaces, time-based routines, interpersonal relationships, and sensory perceptions. The problem we’re trying to solve is that so many of the factors relevant to connected products aren’t easy for people to explain in words.

Words can fail to capture the context of potential use—the spaces, sights, sounds (or even smells!) that will matter to users and to the product. It can be hard for people to adequately describe in words the layout of their homes, or the sensation of hearing a fire alarm next door.

Moreover, psychological research tells us that human perception and memory are not objective. Research participants often volunteer stories of frightening—but rare—crises, but do not mention everyday frustrations that they have long since stopped noticing. Then there are the problems inherent in asking people what to design. Your potential users are expert in their own lives, but not necessarily in the practicalities of the design process or the capacities of emerging technologies.

Humans are also notoriously bad at predicting their own future actions. We may tell companies that we would like to have a refrigerator with a screen in the door...but when faced with the opportunity to buy one, we hesitate. Perhaps the *hypothetical* refrigerator was much more compelling than the reality. Or perhaps, when faced with competing demands on our wallets, an expensive new refrigerator turns out to be less important than a family member's costly doctor bills or an overseas vacation.

Problems of knowledge, memory, and perception are problematic for all user-centered design projects, but especially so for connected products. Those everyday troubles can inspire a great new product—or they can doom your bright idea to irrelevance. That is, if your research participants aren't network engineers, they probably can't explain why their wireless networks are unreliable. They may not even know that their problems listening to streaming music have to do with an unreliable network in the first place.

Drawing on the work of Elizabeth Sanders, a designer and teacher, we can divide these factors up into three categories (Table 5-3):

*Explainable*

What people can articulate in words

*Observable*

Tangible events in the world that researchers can perceive with their senses

*Tacit*

Motivations, sensations, and other mental structures that are implicit in action but need to be made explicit in order to be used in design

It's easy to learn about explainable factors: all you need to do is ask people! We can bring other tangible phenomena, such as technological infrastructures, to light with careful and sensitive observation. But aspirations, emotional associations, and the kind of cultural knowledge we usually take for granted are not so easily seen or heard. To use them to help us define a product, we'll need to prompt or provoke research participants to go beyond what is easy to put into words (Table 5-3).



**TABLE 5-3.** Research techniques for connected products<sup>9</sup>

PARTICIPANTS...	YOU GATHER INFORMATION ABOUT WHAT PEOPLE...	YOU LEARN ABOUT HUMAN CONDITIONS THAT ARE...	WE SUGGEST...
Answer questions	Say Think	Explainable	<i>Elicitation activities</i> , such as map-making, prompt richer and more detailed conversation
Behave normally	Do Use	Observable	<i>Field visits</i> discover and document contexts of use, and provoke unexpected encounters
Make and play	Know Feel Dream	Tacit	<i>Generative methods</i> , such as co-design workshops, create artifacts in order to surface hard-to-articulate values, dreams, and assumptions

### ASKING: ELICITATION TOOLS

One of the best ways to go beyond what is easy to observe or articulate is by having participants document their lives themselves. We can then use that documentation in interviews to *elicit* richer and more detailed explanations:

- *Spatial maps* help us associate everyday activities to places.
- *Timelines* help people tell more detailed and more concrete stories about past events.
- *Diaries and usage logs* help you understand what your users do, think, and feel when you're not around.

### Maps

Despite marketing promises, we can never—and should never—design for *anytime, anywhere* usage. Think about our problem of home security. There will probably be a mobile interface...but to monitor

<sup>9</sup> Adapted from E. Sanders and P.J. Stappers, *Convivial Toolbox* (Amsterdam: BIS Publishers, 2013).

conditions we need components to place around the home. Those hardware components need to work with the actual spaces that potential users live in and move through—not the idealized ones we imagine.

Spatial maps help you understand what those homes and other places are like and the relationships their inhabitants have with them and with one another. Maps are convenient for interviewing because they are familiar. Who hasn't drawn or read a map?

Spatial mapping starts with a question about a physical location. You ask someone to represent a place they know well (say, a home) and then gradually fill in the details that interest you. You might investigate operational context by asking for information about wireless networks and power outlets, or investigate needs and sociocultural context by asking emotional questions such as, "Where do you feel most safe in your home?" You can also ask people to draw movement in space, like a commute or a vacation route. What's useful about these maps is the detail they offer about space and time. What's less useful is that it's hard to ask about nongeographic spaces, such as websites or applications.

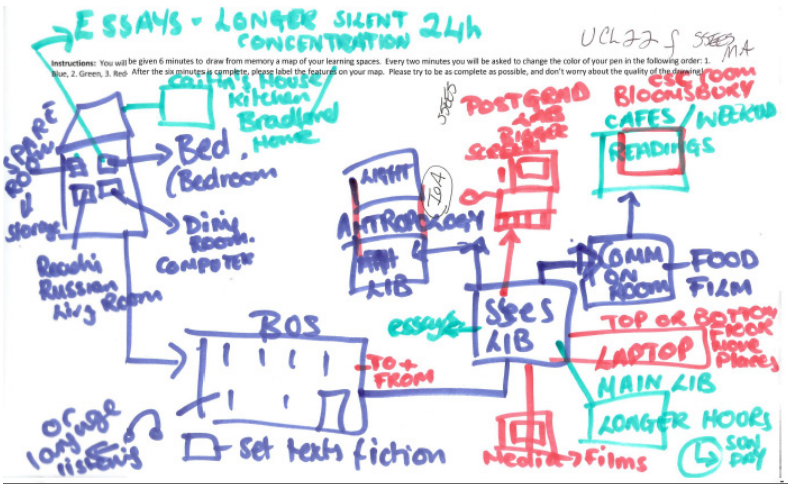
Cognitive mapping (Figure 5-8) starts with a question about an activity. You ask about "securing your house" or "learning to speak French." Those kinds of maps limit themselves to geographic locations, but can include websites, devices, and books. These sorts of maps, anthropologist Donna Lanclos says, help you see "a network of spaces, with a relationship to each other."<sup>10</sup> Cognitive maps often include buildings, technologies, and modes of transportation. However, they tend to be necessarily less informative about any given place.

## Timelines

Making timelines (Figure 5-9) can help participants tell more detailed and concrete stories. Try this right now: write down what you did this morning from the moment you got up to when you left the house (if indeed you left the house). You'll probably start remembering more events as you go, and you'll need to reorder your list as more details come to you.

---

10 For more on how to interpret cognitive maps, see Lanclos, D. (2014, April 17) "Post-Digital Learning Landscapes." Retrieved from <http://www.donnalanclos.com/?p=10>.



**FIGURE 5-8**

A cognitive map from a study of “post-digital learning landscapes” in London overlays types of work on places and devices; it includes sub-areas of a student’s house, her laptop, a commute on the bus, as well as multiple libraries and cafes (image: Donna Lanclos, University of North Carolina at Charlotte)



**FIGURE 5-9**

A timeline helps a participant in a study on energy use walk researchers through a typical day (image: Dan Lockton and Flora Bowden)

Perfect recollection is impossible, but asking participants to write down the steps they can remember on Post-it notes—then having them shuffle and add steps—will jog their memory. Timelines can spark explanations of cause and effect: “I got mad at my mom *so when I left I forgot to lock the door.*” Combining timelines into an overview (Figure 5-10) can surface shared sociocultural values and meanings—information that will help later when you are crafting value propositions.<sup>11</sup>



**FIGURE 5-10**

Combining individual timelines into a single overview surfaces common patterns of engagement with advertising (image: Elizabeth Goodman)

As a practical note, it’s a good idea to leave the question of what events begin or end the timeline up to the participant. Consider our home security project. You may assume that locking up starts when the key hits the lock. Your participants, however, might feel like the process actually starts with a panicked search through the house for perpetually misplaced keys. How to avoid or mitigate that panic might be the goal or need that sparks a new service or product.

11 For a very useful introduction to timelines, see Lockton, D. (2014, January 13) “A case study on inclusive design: ethnography and energy use.” Retrieved from <http://bit.ly/1bD82sF>.

## Diaries and usage logs

Diaries and/or activity logs (Figure 5-11) track activities and goals that happen over time (e.g., learning about a rash of neighborhood burglaries, following the news, changing one's locks). They are just what you think: participant-completed records of what people do, think, and feel over time. You could have participants jot down notes on their improving fitness, for example, emailing you weekly updates. They could supplement what they write with photographs for greater detail. You can even give people deliberately open-ended tasks, such as *Take a photo of something in your house that makes you anxious*.



**FIGURE 5-11**

Diary studies are often in paper form so that participants can scribble and sketch on them more freely (image: Ant Mace)

Those weekly updates are less prone to memory bias than retrospective timelines, because they chronicle events *as those events happen*. When completed, diary entries can prompt interview questions about what participants felt and saw even months earlier.

If you have backend access to a digital product participants already use, you might even be able to get logs of their activity (getting their permission beforehand, of course). Or you can ask them to install an activity tracker app customized to your research questions that tracks the time and place of on-device actions. Those logs will necessarily not be as rich

or evocative as diaries, but you can use post-study interviews to fill in the blanks. And they will definitely give you “ground truth” about what your participants get up to when you’re not around.

## THE IMPORTANCE OF EXPLANATIONS

*While a picture is said to convey more than a thousand words, pictures never tell the whole story. They cannot say “no”; they cannot give reasons; they cannot plan; and they cannot provide feedback on how they are being understood by others.*

—KLAUS KRIPPENDORFF

*The Semantic Turn: A New Foundation for Design, p. 139*

It’s risky to interpret these artifacts on your own. They tend to be ambiguous, with lots of content that is personally meaningful to the participant and not particularly decipherable to a stranger. It’s easy to jump to wrong conclusions. What timelines, spatial maps, and diaries are for, then, is to *elicit* richer and more detailed verbal explanations from research participants.

We’ve highlighted three types of artifacts, but there are many other ways to elicit rich descriptions of current goals, needs, and activities. You can present participants with what Elizabeth Sanders calls a “trigger set” of evocative images from which they choose the most meaningful. You can have them smell, touch, and taste sample materials. You can give them illustrated scenarios of more-or-less futuristic future products to see how they respond. But the goal remains the same: to push participants to go beyond what is easily explainable to what is hard to articulate. It is the objects *plus* explanations that inform design strategy, not the objects alone.

## WATCHING: FIELD VISITS

The most effective way to identify people’s current goals, frustrations, and pleasures is to ask them. That’s logical enough. Interviewing skills such as asking thoughtful questions, being a careful listener, and analyzing the results are the foundation of all user research.

But you’ll want to go beyond verbal questions and answers. Particularly for connected products, user-centered design means taking what people say they want seriously but not always literally. On their own,

interviews will only capture what people can verbalize. The first step is to move beyond the comfort of your own workplace into the places where potential users work, play, eat, and sleep.

Researchers call these trips out of the office “field visits.” If you’re designing security products and services for single-family homes, for example, that obviously means visiting the homes in question. Field visits typically serve two purposes:

- To experience and document concrete “ground truth”
- To engage with difficult-to-articulate cultural meanings

On such a field visit, you might

- *Tour* an entire space (such as a private home).
- *Follow* one person as they perform an activity (such as locking up before leaving on vacation).
- *Observe* all the activities and tools in one specific place (such as the kitchen).
- *Trace* the web of relationships and processes that make an object (such as a police report following a break-in) particularly meaningful to your audience.
- *Document* hard-to-verbalize factors, such as the alarm code scrawled on a piece of paper or an outside door left propped open.
- *Take advantage* of unexpected encounters (such as a home visit on which you meet previously unknown stakeholders, such as neighbors, mail carriers, or forgetful teenagers).

While in the field, your job is to:

### *Seek the unexpected*

The exciting thing about research is that people’s lives are inevitably richer, stranger, and more complicated than you expect. To get the most out of a field visit, you need to seek out those surprising complications. One useful technique is the *AEIOU schema*, developed by business ethnographers at the Doblin Group in the

1990s.<sup>12</sup> In new environments, take 10 minutes to list all the *activities*, *environments*, *interactions*, *objects*, and *users* you can see. Your list will almost certainly give you some new questions to ask.

### *Be nosy*

Being too shy to ask questions negates the whole point of the field visit. Who propped open the door? How did you feel about being robbed? Who pays for the new alarm system? Don't assume that you understand what's going on. Prepare a set of questions in advance, but be ready to keep probing until you get specific, concrete answers. Ask to open closed doors. If you care about health, for example, you'll want to ask (politely!) to see where medicines are kept. Don't assume that participants keep medicines where you would. Don't assume that you and your participants define "medicine" similarly. It should go without saying, of course, never to express criticism or shock about what you discover.

### *Document your experience*

Your memory is not enough. Take photographs, video, and notes of anything that strikes you as relevant and that you'll need to share with other design stakeholders. For example, if you're interested in cooking, then you might ask for a recipe to try yourself. Or you might try to buy some of the ingredients. The shopping trip can be a research exercise on its own, and the results can spark design.

Like so many parts of design, field visits seem straightforward enough, but in practice have enough hidden complications and fascinations to be a life's work in itself. Field visits for design don't have to be the sort of elaborate adventures documented in Figure 5-5. They can also be lightweight encounters, as shown in Figure 5-12.

## **MAKING: GENERATIVE METHODS**

Generative methods, such as co-design workshops, can get nondesigners directly involved in design. In early stages, generative methods help investigate everyday needs, dreams, anxieties, and habits. Later, with a product or service direction chosen, generative methods can help establish how touchpoints should concretely look, feel, and behave.

---

12 C. Wasson, "Ethnography in the Field of Design," *Human Organization* 59:4 (2000).





**FIGURE 5-12**

“Water me” signs sighted at a neighborhood-run garden inspire initial service proposals for collective urban resource management (image: Elizabeth Goodman)

Co-design workshops are semi-structured sessions in which professionals ask potential users to design new products and services by themselves, for themselves. Participants typically get kits of basic components they put together to make artifacts that tell stories or represent design concepts. Common outcomes include paper collages (Figure 5-13), or even LEGO models (Figure 5-14). Those artifacts, like objects in elicitation exercises, can then prompt rich, nuanced, concrete discussions about what people want for their lives.



**FIGURE 5-13**

Participants in a co-design exercise stick paper cards representing different types of functionality, such as a calendar and a keyboard, to a real car’s steering wheel and dashboard; note that the existing features are covered up to provide a blank canvas (image: SonicRim)



**FIGURE 5-14**

LEGOs help workshop participants imagine and act out telecom service scenarios (image: Andy Polaine)

Design consultancy Experientia uses generative methods to:<sup>13</sup>

- *Draw out* intangible aspects of the product’s environment, such as ethics, social norms, or cultural values—particularly ones that demand a choice among conflicting options
- *Generate resolutions* for those conflicts, particularly processes over time that engage multiple stakeholders and touchpoints
- *Free participants* from unnecessarily conservative or limited solutions, opening them up to imagine more diverse and more unpredictable future products and services
- *Anticipate* how habits and behaviors might change with new products and services

Why use generative methods? The role of research is straightforward, if logistically sometimes complicated, in updating an existing product or designing a new entrant into a well-established product category. You can often draw a clear line from what you have observed to what you design—from observations, to insights, to principles, to definitions

---

<sup>13</sup> “Creating Together: Building Value with Participatory Design,” Experientia, <http://bit.ly/1bD8QoK>.

(see Chapter 6 for more). If you're trying to invent an entirely new product category, or create an entirely new entry into an existing product category, user research gets more conceptually complicated. Along with information about existing practices, you need *inspiration*—grounded in everyday life—to invent something completely new.

Generative methods, especially co-design workshops, are one way to inspire fresh ideas that are still deeply grounded in people's dreams and aspirations. They are also good ways to provoke sociocultural discussion and debate to inform prototyping.

## Summary

Research doesn't eliminate risk or creativity.

Successful products and services that require cooperation (or at least, not active rejection) depend on a web of relationships among direct actors, indirect stakeholders, and the contextual factors. While users are central, designers ignore nonusers and product context at their peril.

Connected products and services thrive when they acknowledge people's diverse *roles*, *expertise*, and *needs*. Most obviously, there are immediate *actors* who make, use, repair, and dispose of the product. However, indirectly involved *stakeholders* also affect how the product works. Given such diversity, needs and goals are likely to be in conflict. Even if you can't resolve any conflicts, your job is to anticipate them.

The success of connected products and services is also tightly coupled to contextual factors. *Operational factors* are those necessary for a product to function at all, which usually means appropriately engineering hardware for its environment but can also include the availability of replacements and parts. *Behavioral factors* refer to human physical activity and states of being. *Ecological factors* are the networks of organizational, legal, and economic relationships that support operation, such as third-party application developers. Finally, shared *sociocultural factors* inform how people make sense of the product—and how often and why they reject it.

Research should go well beyond asking people what they want. Enriched with diaries, maps, and timelines, *ask* activities elicit rich descriptions of actions, attitudes, and desires. *Watch* activities investigate nonverbal,

contextual factors, often by sending teams on field visits. Generative *make* activities surface hard-to-articulate values, dreams, and assumptions by creating tangible artifacts.

Building a connected consumer product without upfront investigation will likely waste your time and money. However, we're not advocating a traditional waterfall model, in which you do extensive research, build the product, and only then evaluate it. That's likely to waste time and money, too. It's simply impossible to anticipate every crucial question before you have a prototype.

Instead, we're advocating what researcher Erika Hall calls a "just enough" approach. Do *just enough* investigation to answer the initial questions, then start quickly prototyping, then go back and use your prototypes to learn more about your audience and their world.

In the next chapters, we'll discuss how to do that.

# Translating Research into Product Definitions

**BY ELIZABETH GOODMAN**

So you did some research. You have piles of observations, images, video, and notes. Now what?

Designer Bill Buxton sums up the next steps in two phrases. You need to make *the right thing*—that is, make something the target audience finds valuable. You also need to make *the thing right*—that is, make the product desirable and usable. To do both of those requires translating the raw data from your research into design action.

This chapter is intended to help you take on two challenges to *making the right thing*.

The pressing need for clear communication and decision making is the first challenge. As we said in Chapter 4, connected products often rely on complex webs of other products, services, and infrastructures. Mass-market consumers, in areas in which they do not have deep technical or domain knowledge, generally expect a product to come designed and engineered to fulfill a specific need. So the benefits of connected products are often hard to communicate succinctly and clearly...not just to the buyers or users of the product, but also to the people building it.

Frequent friction between business models, users' expectations, and shared values form the second challenge. This friction can lead not only to UX problems, but also very public controversies if (and more likely when) users discover them. We believe that translating engagement with stakeholders directly into product definitions is the best way to anticipate potential mismatches, and then to either mitigate them or turn them into advantages.

This chapter introduces:

- Product definition techniques for analyzing values and benefits (page 188); explaining what your product is (page 195); and deciding how it works (page 203)
- Common mismatches between business models, technical capacities, and user expectations (page 207)

It also addresses the following issues:

- Why you should expand your definition of “value” (page 187)
- Why metaphors are important—and dangerous—in product definitions (page 195)
- How to generate and use design principles (page 199)

This chapter outlines important concepts and questions for translating your research into definitional statements that tap into what stakeholders care about. Along the way, we’ll also discuss how tools you may already use, such as stakeholder profiles and customer journey maps, might help you answer those questions.

In Chapter 14, we’ll discuss *making the thing right*.

## Generating the Elevator Pitch

Our goal in this chapter is to help you put together the “elevator pitch”—a concise statement that communicates:

- What the product is
- What the product does
- And why the product is important

From full-on value proposition documents to glossy renderings, there are many ways to define a product in the early stages of design. But the elevator pitch should be the starting point for all of them.

Most importantly, writing an elevator pitch forces critical thinking. If you cannot succinctly explain what your product is and how it could work, you’re probably avoiding important questions of usefulness and feasibility. Until you’ve answered those questions, you shouldn’t be spending time and money making beautiful images.

You also need a one-sentence explanation for a very pragmatic reason: that's all the information people new to the project generally want to hear.

Let's structure it as a fill-in-the-blank exercise:

*Product name* is a(n) *type of object* that helps *category of people* name of *job to be done* by *technical means*.

You can move around the elements to make a nicer sentence, of course. For example, take this statement from energy information provider Opower:

Opower combines a *cloud-based platform*, *big data*, and *behavioral science* to help *utilities around the world* reduce energy consumption and improve their relationship with their customers.

At this point, we hope you have filled in the *category of people* blank already—that's where you started your research, after all. This chapter focuses *the job to be done* and *type of object* parts of a product definition. We'll discuss the *technical means* component only in how it affects the job at hand. The next set of chapters will take on the *technical means* part in more detail.

## Why Does Your Product Matter?

Chapter 4 introduced the idea that value propositions should drive UX. Actors need a clear sense of what the product might do for them and why they might want it (see Figure 4-2) despite the costs associated with buying, installing, and using it. If the value proposition doesn't make sense to purchasers or users, no amount of well-crafted interaction or industrial design will make the product successful. A value proposition supplies the “why is this product important” part of an elevator pitch and helps fill in the *job to be done* blank.

There are many references you can consult to learn about value propositions in general. What we're interested in here are the questions and opportunities that arise from new technical capacities for data collection, sharing, analytics, and automation.

Often, basic research presents us with a straightforward value proposition. What immediate problems do people face? Why aren't their current solutions working? How could you provide something better? And

often, the next steps are quite clear: you provide a product that helps people fix their problem at a price they are prepared to pay, and everyone wins.

This section will focus on harder questions of product definition. What happens when the benefits and costs envisioned by multiple product/service stakeholders conflict? As with the conflicts over needs we discussed in the previous chapter, tensions among benefits, costs, and values have high stakes and are surprisingly common. As lawyers like to say, *cui bono*? In other words, who benefits?

Take the Target “pregnancy coupon” controversy.<sup>1</sup> Target, a retailer based in the United States, assembled a dataset combining female customers’ purchases and information on customers from outside sources. Using Target baby gift registries, the company developed a predictive model of likely pregnancy based on the long-term buying patterns of women who created a registry. Then it started sending coupons to women whom the model calculated were probably pregnant.

At that point, someone within Target realized that the coupons were, in the words of reporter Charles Duhigg, a “public-relations disaster” in the making. A pilot run showed that many customers did not react well. It’s easy to see why. Pregnant recipients could perceive the coupons as an unwanted corporate intrusion into personal matters. Recipients who were *not* pregnant might find the suggestion of pregnancy offensive. Or worse, the coupons might be hurtful: becoming pregnant (or not being able to become pregnant) is often an emotionally sensitive topic. As Duhigg writes,

So the question became: how could they get their advertisements into expectant mothers’ hands without making it appear they were spying on them?

What Target did was mix pregnancy-related coupons with unrelated offers. That way, the pregnancy coupons appeared to have been included by chance, making customers less likely to take offense from them.

---

<sup>1</sup> C. Duhigg, “How Companies Learn Your Secrets,” *The New York Times*, February 16, 2012, <http://nyti.ms/1bDaobf>.



Clever? Or creepy? Customers don't voluntarily *choose* to be a part of this program: all Target shoppers are automatically included and can't opt out. Clearly, many customers don't want Target to investigate their reproductive status. Equally clearly, Target wants to lock in a lucrative business in diapers and baby goods *before* the baby arrives. Sure, the customer will save some money with the coupons. And they don't have to do anything special to participate. But are the tangible benefits worth feeling like a giant multinational corporation is snooping into your private life? Does the situation really improve if the company conceals the predictive analytics component of the program, so customers don't have the opportunity to object?

Over the rest of this chapter, we will map some of the complexities of defining "value" when it comes to mass-market connected products and services.

### **Expanding the definition of "value"**

The traditional definition of "value proposition" takes "value" to mean the *tangible benefit* (saving money, for example) the product delivers. But that definition, while straightforward, doesn't quite account for the controversy we just described. The coupons saved both money and time—yet many people did not see the value. In considering "value" during product definition, then, we should expand this usual definition to include intangibles such as personal aspirations and moral codes of behavior.

**Aspirations and goals.** What we value is what we care about. Often, we care about things we want to do, be, or have but are not likely to achieve. That's the true value proposition behind most "affordable luxury" products and services: they allow purchasers to feel closer to their dreams. After all, most of the people who buy expensive athletic gear aren't actually going to be climbing Mt. Everest or training for an Iron Man triathlon. If your business model involves helping people feel closer to specific dreams, it's worth taking time to clearly articulate what those dreams *are*.

**Moral behavior.** What do stakeholders believe is the *right* way to act? What we believe is right affects everything we do—down to the products we buy and the services we use. You don't need to share your users' morality, but at the very least you need to take it into account. The

pregnancy coupons bring up two questions of “right” behavior. The first is the behavior of the recipient—pregnancy (or, more specifically, the activity that leads to pregnancy) is morally loaded for many people. The second is the behavior of the coupon-issuer: What kinds of data collecting and modeling do consumers expect from a retailer?

Seemingly straightforward economic decision making often triggers these questions. The moral value of privacy—which we could define in this situation as “security from intrusion into one’s intimate affairs”—is a foundational value for Target’s core customer base in the United States. Which is why, as we discuss in Chapters 11 and 15, the right to privacy is enshrined in law not just in the United States but in many other countries. What Target did could be interpreted as an unwarranted violation of that value—that is, immoral spying. How willing are you to take actions that may violate what your users believe to be right?

Whatever you think about Target’s decision, they did one thing right: they confronted the looming conflict between what they wanted and what their customers wanted head-on. They did not pretend it did not exist, or that it would go away if they did nothing.

As the Target example showed us, wearables, medical devices, and home sensing products are only the most obviously value-laden products. Even a credit card or email address can spur debate. The more a product or service touches on conditions and behaviors that users can reasonably assume are private, the more likely you are to face questions like Target’s. At the end of this chapter, we’ll outline some of the most common concerns.

## **TOOLS FOR DISCOVERING AND COMMUNICATING VALUE**

When you’re defining a product, you need to be clear about where your definitions of “value” or “benefit” come from, and to whom they apply. Research tools can help you get a handle not just on what people care about—but where their goals and your own goals might collide. Technology policy scholar Katie Shilton calls these sorts of activities and documents *values levers*<sup>2</sup>—tools for articulating what stakeholders value more explicitly, drawing out consequences, and negotiating conflicts.

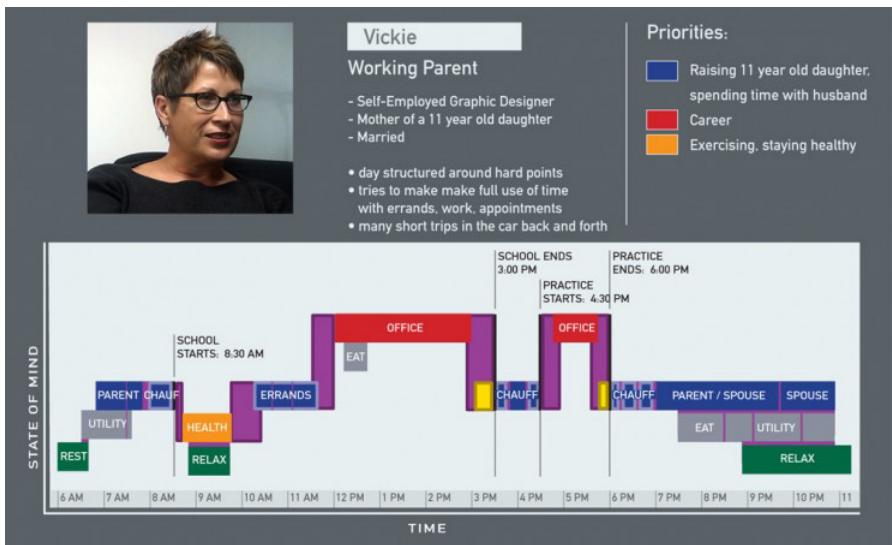
---

2 K. Shilton, “Values Levers: Building Ethics into Design,” *Science, Technology & Human Values*, doi:10.1177/0162243912436985.

**Portraits.** A portrait is just what it sounds like—a representation of a person, typically a category of user. Portraits make observed values and behaviors more concrete and more relatable in order to better support decision making.

The best-known type of portrait is the persona. Personas are fictional characters synthesized from recurring patterns and compelling stories found in research. So though personas are not real people, they draw upon real research data. Persona advocates praise them for making massive amounts of data relatable and usable. Contrasting personas can be used to highlight attitudinal and behavioral differences that are present, but less obvious, in the data. Detractors point out that personas are, well, *not real*, and that practically it's very hard to make memorable yet realistic ones.

There's another option, of course: if you want to bring an understanding of real people into your project, why not just use data from the real people you met? That's what many designers do. They distill representative quotations and images from participants into brief profiles (as in Figure 6-1) that illustrate research themes.



**FIGURE 6-1**

A profile of Vickie, a working parent (image: Karten Design)

Whether you use personas or profiles, portraits can identify, make visible, and then negotiate potential value and need conflicts in the course of developing a product definition:

### *Prioritization*

No product can be all things to all people. We keep on making this point because feature bloat continues to be a problem. Using portraits to map out important dimensions of difference among people is a first step in deciding what, and whose, needs or values you will prioritize. It also helps you figure out how (or if) to compensate those you ignore. Shilton advocates naming and labeling certain moral values, such as “privacy,” “freedom,” or “justice,” to make them more available for discussion. If your project involves open source technology, for example, commitment to “openness” (being explicit about what it concretely means) should show up in your portraits. If an important group of users is committed to openness defined as “open source development,” and your business model depends on proprietary code, portraits can help teams come to grips with potential tensions.

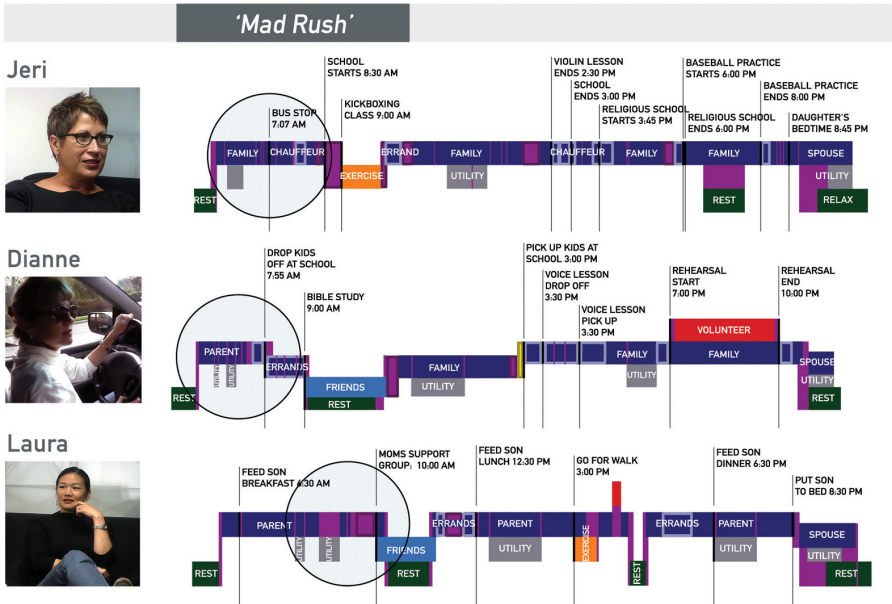
### *Generalization*

Highlighting similarities among different groups of people can help you form design principles (see next section) to support the widest slice of people. That is what Karten Design, a product innovation consultancy, did in studying “time empowerment” for Johnson Controls: it found that different types of people used time in the car for relaxation. The firm was then able to conceptually recast that time in the car as a valuable “oasis” of rest amidst a flurry of activity (Figure 6-2).

### *Inspiration*

The inspiration for great products doesn’t always come from happy stories about middle-of-the-road users with easily met needs and conventional values. What about people at the extremes, like early adopters, intense enthusiasts, and skeptics? Early adopters are probably unlike latecomers in important ways. But enthusiastic early adopters can tell you something about what it’s like to live with a new type of product everyday. Skeptics are also useful—they can make you consider how your ideas might violate certain values and expectations.

## Vehicle Interior Innovation: *Time Empowerment*



**FIGURE 6-2**

Comparisons of the activities and emotions experienced by parents with different relationships to time management (image: Karten Design)

### *Aspiration*

Portraits are usually realistic representations of what groups of people are actually like. But they can also represent dreams and fears—which are hard to see in videos but which loom large in our lives. Portraits can illustrate what users want to be. Or what they fear becoming. Extreme portraits can help define aspirations even if the probable user is more middle of the road.

### **Immersion tools**

Decades of cognitive science have shown that we learn differently when we *do* an activity from when we watch it being done. Through semi-structured activities—having people try unfamiliar tasks, play games, or wear costumes, we make diagrams and other static models

both immediately meaningful and memorable in the long term. Design researcher John Payne calls such planned active engagement “immersion tools.”<sup>3</sup>

Often, teams use *simulation exercises* to evoke the experience of a target audience. *Games* with clear rules are a way to convey the logic at play decision points and trade-offs (Figure 6-3). Costumes or props, such as the Empathy Belly used by Ford (Figure 6-4), can induce physical or cognitive states radically different from those of the design team.



**FIGURE 6-3**

Playing a “card game” helped ground product brainstorming in the research (image: John Payne and Alexa Curtis)

Many people describe this immersive understanding as “empathy.” But “empathy,” we think, is better understood as an ideal rather than a predictable outcome. To empathize is to feel just what someone else feels. Can you really feel what it’s like to live as a paraplegic for the rest of your life by using a wheelchair for an hour? No. Will playing a card game about border crossings really submerge you in the worldview of a migrant worker? No. However, immersion tools *can* help you gain a less abstract, more concrete sense of the decisions and sensations of

3 J. Payne. “From Experience Models to Immersion Tools: Transferring Ethnographic Knowledge In An Agile World,” August 13, 2014, <http://epicpeople.org/blog6/>.

everyday life. And that will give you a leg up in crafting value propositions that take into account intangibles such as morals, aspirations, and fears.



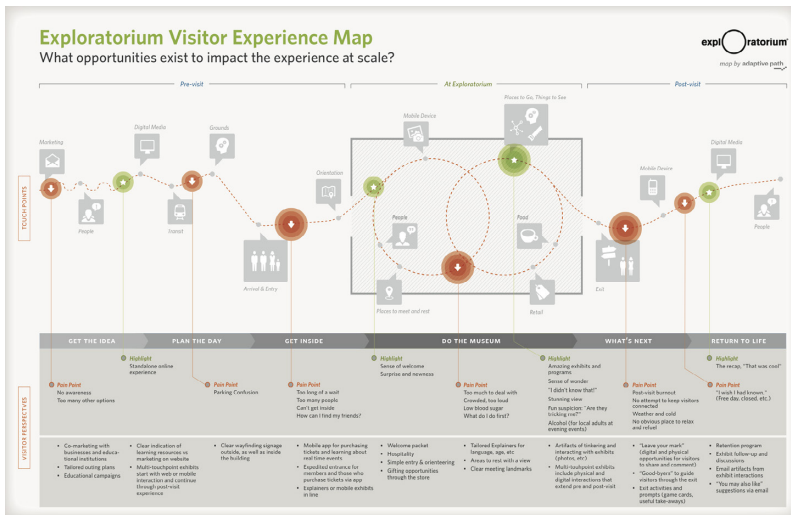
**FIGURE 6-4**

A Ford engineer demonstrates the use of an Empathy Belly to simulate the sensations of driving while pregnant (image: Ford)

**Customer journey maps.** A customer journey map (sometimes called an experience map or experience flow) represents how an actor interacts with a product or service over time. A journey map includes places, events or time periods, touchpoints, and interactions. Karten's profile of Vickie (Figure 6-1) contains a diagram that looks like a customer journey map—in this case, the “journey” is a typical day. Customer journey maps, however, more typically center on coherent activities such as a museum visit (Figure 6-5). They are key tools for identifying what jobs customers might need done at specific points in time—and why those jobs matter.

## What could a customer journey map include?

- Important people and organizations (from the perspective of the customer/user)
- Key locations (i.e., “the car” or “the hospital”)
- Tools and technologies in use (paper counts!)
- Activities, processes, and interactions over time
- Touchpoints where the user encounters the service
- Pain points and highlights (from the perspective of the customer/user)
- Opportunities for design-led changes (often identical to pain points)



**FIGURE 6-5**

Map of museum visitor journey by design consultancy Adaptive Path for the San Francisco Exploratorium; orange circles mark pain points, green circles mark highlights (image: Adaptive Path)

Journey maps are particularly useful for designing connected products and services because they put the *experience*, not a technology or task, front and center. In the previous chapter, we defined context as all the factors *outside* the product that become relevant as people engage with



it. In showing activities over time, the journey map can explain where and how contextual factors become relevant (or irrelevant). They help teams see experience as evolving, rather than static.

## What Is Your Product?

Knowing what people value, how do you decide what *type of object* you're designing?

Let's return to our original fill-in-the-blank product definition:

*Product name is a(n) type of object that helps category of people name of job to be done by technical means.*

Making conceptual models is the next step in filling in the *type of object* blank.

### CONCEPTUAL MODELS AND DOMAIN METAPHORS

In the words of Don Norman, a conceptual model is “a story that puts the operation of the system into context: it weaves together all of the essential components, providing a framework, a context, and reasons for understanding.”<sup>4</sup> The first two words are the most important part of this definition: a conceptual model is *a story*. And metaphors are one of the best ways to tell a story.

Metaphors work by linking two unrelated concepts together, like “watch” and “coach.” The classic example of a conceptual model as metaphor is “the desktop.” For the past 40 years, the components and organizational logic of a 1960s-style office desk have supplied the underlying metaphor for interfaces to computer operating systems...long after the 1960s-style office itself disappeared. Some more recent examples:

- The Nike Sportswatch: a “personal running coach”
- The Nest: a “Learning Thermostat” (with “nest” itself a common metaphor for a home)

Sometimes metaphors are direct: the watch is a running coach. Sometimes the metaphors are implied: the Nest is a student who is “learning.” Such metaphors help make the unfamiliar, such as a watch that tracks your movements, seem more comfortably familiar.

---

4 D. Norman, “Design as Communication,” <http://bit.ly/1bDb5kG>.

Before deciding what type of object you're making, consider how conceptual metaphors might be shaping your thinking. Scholars of metaphor and language argue that we don't just speak in metaphors, we "live by" them. Metaphorical comparisons shape thought and action.<sup>5</sup> For example, the metaphor "Argument is war" produces statements like: "Your claims are *indefensible*" or "He *attacked* every weak point in my argument." As scientists George Lakoff and Mark Johnson point out:

The metaphors not merely in the words we use—it is in our very concept of an argument. The language of argument is not poetic, fanciful, or rhetorical; it is literal. We talk about arguments that way because we conceive of them that way—and we act according to the way we conceive of things.

One common recommendation is to match the product's *conceptual model* as implemented by designers to its users' *mental model*. Mental models are expectations about what the product is and how it works. They are acquired over time, through use and education. If the product's conceptual model doesn't match up with users' mental models, the product will at best be difficult to use. The 1960s office, for example, may now be an actively confusing conceptual model. Its follow-on interaction models (see Figure 4-2) of static hierarchical folders and the trash can notoriously mislead inexperienced users about the functions of aliases, dynamic folders, nonpermanent deletion, and other important tools for digital information organization.<sup>6</sup> (Conceptual models in the context of IoT systems are discussed in Chapter 9.)

But there's a necessary step before modeling the product that is often forgotten: modeling the *domain*. The stories we tell about products necessarily descend from the metaphors we use to explain situations. So reuse of domain metaphors is dangerous when it leads you to transplant (see, another metaphor!) your own values and aspirations to inhospitable climates. If your domain metaphor conflicts with users', the product conceptual model and interaction models likely will conflict as well.

---

5 G. Lakoff and M. Johnson, *Metaphors We Live By, Second Edition* (Chicago: University of Chicago Press. 2003).

6 A. Katifori, G. Lepouras, A. Dix, and A. Kamaruddin, "Evaluating the Significance of the Desktop Area in Everyday Computer Use," 2008 *First International Conference on Advances in Computer-Human Interaction*, (2008): 31–38, doi:10.1109/ACHI.2008.27

Take the long history of “home automation” products. The conceptual model is that of *programming*. The follow-on interaction model implies connecting inputs to outputs as if wiring up a machine. Both models, as designer Mike Kuniavsky points out,<sup>7</sup> draw on an underlying domain metaphor: *your home is a factory automated by robots*.

The problem, to revisit our discussion in Chapter 4, is that the available evidence indicates that treating homes like factories leads to frustration for product makers and users alike. Lots of people don’t want a home life with qualities associated with factories, such as rigid routines, constant labor, and boredom. It doesn’t matter if factories don’t actually have these problems—what matters is the conflict between what customers value about homes and what they imagine factories to be like.

Moreover, mental models based on imagined factories can mislead inexpert users about what the systems will really do on their behalf. Take the traditional “automated” sprinkler system. Is it really automatic? Yes and no. Rain or shine, the program follows the same schedule, potentially drowning already rain-soaked plants or letting parched plants wither. Connecting the system to sensors and weather reports can make the schedule more sensitive to environmental conditions. However, different plants have different watering needs. You cannot place a rose and a cactus on the same irrigation “assembly line” and expect them both to thrive.

Like a *real* factory, “automated” watering systems require lots of tinkering and monitoring to cope with unexpected events and variable materials. They require some knowledge of plants’ needs, and the skill to diagnose plants’ problems. Poor results (and hence difficulty in acquiring and keeping customers) can come from the implicit “set it and forget it” promise of the “automation” metaphor.

And, most importantly, it’s by now very clear that the follow-on interaction model of “programming” only appeals to a small subset of the market.

---

<sup>7</sup> You can read much more on metaphors and connected products in M. Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design* (Burlington, MA: Morgan Kaufmann, 2010) Elsevier.

So as you're filling in the *type of object* blank, ask yourself: what kind of domain metaphors might be influencing your conceptual model? What other metaphors might be closer to how your audience sees the world? What other metaphors might be more creatively fruitful?

## GROUNDED INNOVATION

We aim for *grounded innovation*<sup>8</sup>—product definitions that explore new territory while staying firmly rooted in existing activities and values. Ideally, we get to start with values (both tangible and intangible) and brainstorm ways to further them. Your target users aspire to playfulness and creativity: *A home is an orchestra*, and your product is the conductor's baton for an ensemble of devices. They want to feel like their house is alive: *A home is a garden*, and your product, like the Roomba, is an insect that might not be as smart as a human but helps the garden thrive.

However, much of the time designers start with the basic definition already established. If you work for an automobile company, don't try asking to design a bicycle helmet. In a technology-driven project, a grounded innovation approach might start with what a platform can do and work backward. Home automation products are still *un*-nuanced and *un*-adaptive compared to humans. But that doesn't mean owners can't enjoy having them around.

Consider the Roomba. Many Roomba owners regard their little robots fondly, like members of the household. Roomba owners probably aren't your target audience. However, their pet-like or familial relationships with their Roombas can serve as a useful analogy. What happens if you view a home automation system as a pet? A home automation product probably won't even seem as intelligent as a puppy. But could the attachment of owners to less intelligent pets, such as spiders or snakes, inspire home automation design?<sup>9</sup> Metaphors don't always have to derive from your immediate target audience—you can draw them out from analogous populations and situations as well.

---

8 This phrase comes from L. E. Holmquist, *Grounded Innovation: Strategies for Creating Digital Products* (Burlington, MA: Morgan Kaufmann, 2012).

9 *Grounded Innovation* contains helpful case studies and detailed instructions on using analogies and metaphors in designing new types of products, including the inspiration for the robot-as-pet example.

## What Does the Product Do?

Finally, we consider *technical means*. “Means” here includes:

- Abstracted imperatives, or design principles, that prescribe how the product should work
- Specific proposals for the concrete service elements and that will implement those abstracted principles, often generated by service ecology maps

One way of looking at functionality doesn’t necessarily have to come before the other. Design principles can guide how teams use service ecology maps; mapping activities can motivate design principles. What’s important is using these and other techniques, such as journey mapping, to ground technical decisions in the foundational values and metaphors you derived from encounters with users and other product stakeholders.

### DESIGN PRINCIPLES

Design principles are concise imperative statements. They translate values and metaphors into action by helping teams generate and evaluate design ideas.<sup>10</sup> If conceptual models provide high-level inspiration, design principles guide action.

Here’s one from Opower (all of the Opower design principles are available at <http://opower.com/designprinciples>):

Always lead to action.

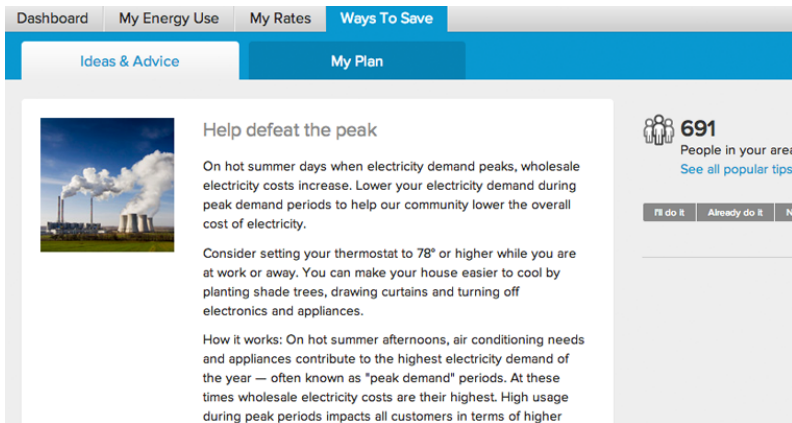
Or another, for a British home security maker:

Be calm. Don’t bug me.

Design principles help translate research themes into action because they are loosely prescriptive. They are loose, in that they can apply to multiple interfaces and media (see Figures 6-6 and 6-7). They are prescriptive, in that they require—or discourage—specific results or methods.

---

10 For a vast number of examples, visit <http://www.designprinciplesftw.com/>.



**FIGURE 6-6**

Example of solution generated from “Always lead to action” Opower design principle: a web page with steps to save energy at peak use times (image: Opower)

Good design principles help you design better connected products and services when they:

*...are clear*

Deciding upon design principles should be a collaborative exercise in clarifying your assumptions and the evidence for them, then agreeing on how to respond when interests collide.

*...are open*

Useful principles leave the specific form of the solution to the designer. “Always lead to action” can motivate both website content strategy (Figure 6-6) and the creation of a simple paper doortag (Figure 6-7).

*...are agreed upon*

Making design principles can help groups of people come to agreement about what is important to them, and why. But principles that aren’t honored across an entire organization are sources of conflict, not of consensus.

*...are authoritative*

Taken seriously, design principles can rally a team to support hard decisions. They can help maintain consistent tone and interaction mechanisms across disparate interfaces. They can help you decide

which devices should enable which activities (see Chapter 9 for more). Most importantly, they can help groups say “no” and *mean it* to features and attributes that aren’t necessary.



**FIGURE 6-7**

Example of solution generated from “Always lead to action” Opower design principle: a paper door tag to help people remember to actively manage thermostat settings (image: Opower)

Opower is just one successful company that uses design principles to both generate service ideas and evaluate them. According to Opower’s Deena Rosen, Suelyn Yu, and Ryan Devenish:

The principles help us generate ideas by framing brainstorming around selected principles in the exploration phase. Later in the process, once concepts are further developed, we check them against each principle. They provide a common framework for anyone to evaluate ideas across the product team. For example, during a design critique you might hear:

“Using principle number two *assume people don’t care*, I prefer concept A over concept B because it creates a stronger emotional response that can grab the user’s attention.”

There’s no single method to generate design principles. Opower, for example, derives its design principles from behavioral science. The company believes that rigorously tested findings are more likely

to remain valid across projects and populations. On the other hand, the design firm IDEO drew “poetic” design principles like “1+1=3”—or “build something new out of two things that don’t usually belong together”—for the Brazilian accessory maker Havaianas from casual observations of Brazilians’ improvisational tinkering.<sup>11</sup>

To test the quality of a principle, ask:<sup>12</sup>

- Does it help you say “no” more often than “yes”? Design principles should help you cut down your options, not increase them.
- Does it help distinguish your product from the competition? If not, then it’s probably not important enough to turn into a general principle.
- Would you ever aim for the opposite? “Be usable” seems like a good principle—but nobody ever aims to make an unusable product. Design principles should be specific to your project, not universal truisms.
- Is it specific to the product or company? Opower derives its imperatives from the psychology of energy use. The IDEO team visited the Brazilian base of its client to inspire accessories for a global market. Both strategies generate useful principles by grounding abstract imperatives in the immediate, concrete context of design and use. If an otherwise sound-seeming principle isn’t tightly linked to the business or product, perhaps it’s more of a general recommendation.

Design principles aren’t written in stone. You should keep testing them as you go, revising with new research findings and new product goals. We give one simple template for generating design principles in Table 6-1.

---

11 J. F. Suri, “Poetic Observation: What Designers Make of What They See,” in *Design Anthropology*, ed. A. J. Clarke (Vienna: Springer Vienna), 16–32.

12 Derived from J. Spool, “Creating Great Design Principles: 6 Counter-intuitive Tests,” <http://www.uie.com/articles/creating-design-principles/>.



**TABLE 6-1.** Using design principles to generate and evaluate service design propositions (framework adapted from a template by Traci Thomas)<sup>13</sup>

	ANALYSIS		DESIGN		
	THEMES	FRAMEWORKS	PRINCIPLES	CONCEPTS	TESTS
Intentions	Descriptive	Descriptive	Prescriptive	Generative	Evaluative
Questions to ask	What are the patterns or key themes in the data?	Why is this pattern important? Why is it occurring? What are some usual consequences?	What should the product do in response?	How might we...?	Should we do this?
Examples	People often forget even simple ways to save energy	To save energy, utility customers must change their behavior. Users should always get a clear way to do that.	Always lead to action	<i>Door tag for in-situ reminders</i> Website highlights concrete tips and advice	Does Concept A lead to action? No? Then we won't implement it.

## SERVICE ECOLOGY MAPS

Underlying the UX is a core question: *what are the basic elements of the whole system and how do they fit together?* Most technical decisions will be made in iterative phases of design, development, and evaluation (see Chapter 14 for more). For the purposes of initial product definition, however, you need some reasonable guesses at the sorts of technologies involved.

For example: Does your product use *reprogrammable, sticky sensors to monitor domestic activities and conditions?* That would be the Mother smart sensor system. Does it use *body-worn sensors and a smartphone app* to help you improve your physical performance? That would be the Nike Fuelband.

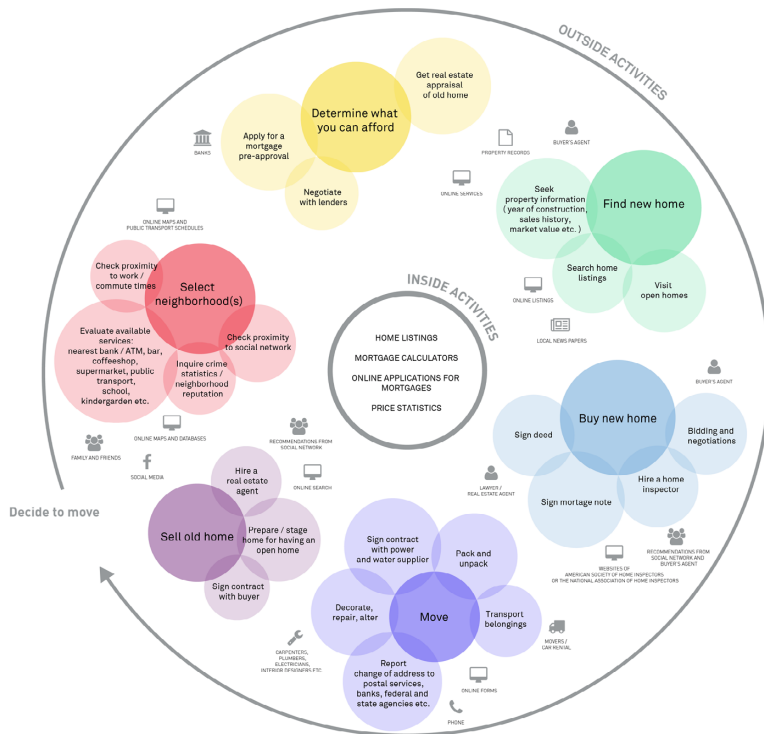
13 T. Thomas, "A Simple Framework for Generating Insights," <http://bit.ly/ibDdFXO>.

A simple-seeming device is often just the most visible part of a complex service-product system. The GlowCap, for example, is a service that reminds users to take medication on schedule and tracks compliance. First, the pill bottle top senses when it is being opened. If the user doesn't open the bottle on time, it glows and makes noise. If the audio-visual reminders don't work, the patient receives a call or text. The service can notify a caregiver automatically, order refills, and send a weekly report to a clinician or caregiver. The value proposition is admirably clear: just-in-time reminders and automated reporting improve medication compliance and hence health outcomes for anyone who takes medically necessary pills.

Implementing that simple idea is complicated. Just refilling a prescription through the GlowCap requires handoffs between five different telecommunications systems. The bottle cap (battery powered) has only enough power to connect (1) to a mains-powered base station, which in turn sends data to the servers via the subscriber's cellular network (2). The Vitality server requests an automated phone call (provided by a third-party company) to confirm the refill with the user (3). Once confirmed, the Vitality server sends a refill request message to the user's pharmacy (4). Once the refill is ready, the pharmacy contacts the user (5).

Clients for that medication compliance data include pharmaceutical companies, retail pharmacies, and health care providers. All that health data storage and transmission must follow national and state privacy laws. Moreover, each client has a different interest in the data, and wants a different view of it. The healthcare providers get printed monthly updates on patients, and potentially integration with electronic health records. The GlowCap is much more expensive than a nonnetworked bottle cap—who pays for the difference? And how might pill-takers acquire a GlowCap? Directly from the pharmacist? From a prescribing doctor? On their own? What happens when the patient no longer needs the GlowCap? Is it recyclable?

A service ecology (or ecosystem) map (Figure 6-8) helps answer questions like these. It translates research into a legible diagram that represents the most important elements and how they are connected.



**FIGURE 6-8**

An ecosystem map for a fictional online home-listings company. It labels the company’s functions and services as “inside activities” and the activities that users handle themselves as “outside activities” (image: Sofia Hussain)

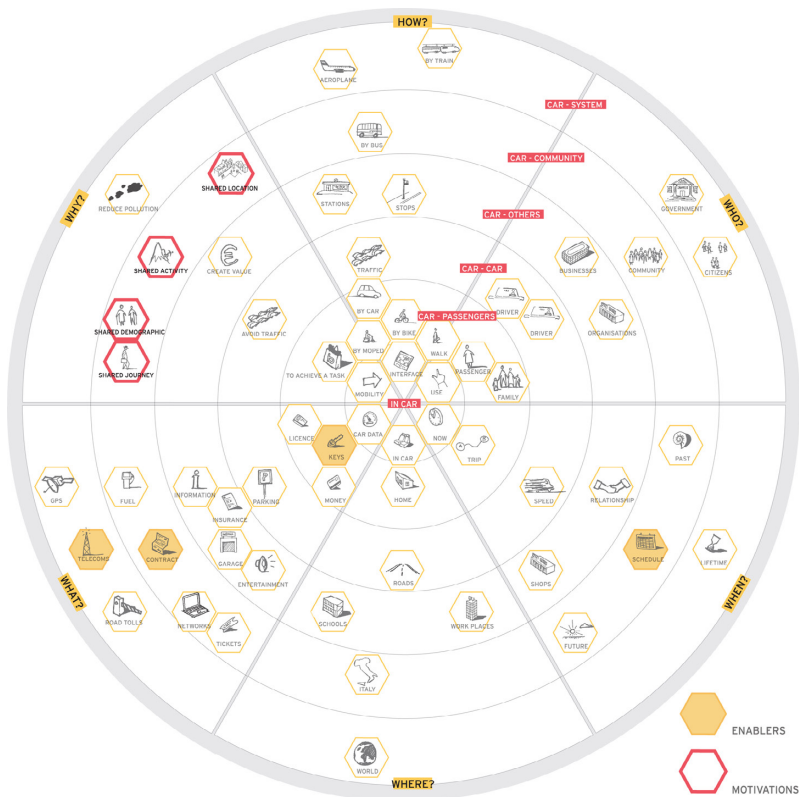
Service ecology maps:<sup>14</sup>

- Give you a way to keep track of relevant actors and stakeholders
- Surface relationships among those actors and stakeholders
- Help you reimagine how those actors and stakeholders might relate to each other

Importantly, service ecology maps can visualize people, organizations, and even places that *are affected by use but are not users*. This is particularly useful for complex domains such as health, finance, or

<sup>14</sup> This list, and Figure 6-11, drawn from A. Polaine, L. Løvlie, and B. Reason, *Service Design: From Insight to Implementation* (Brooklyn, NY: Rosenfeld Media, LLC, 2013).

transportation, where there might be regulatory agencies, municipal or national laws, and unions involved. But it can also help you grasp even the ordinary complications of running a household, as in Figure 6-9.



**FIGURE 6-9** An ecosystem map for a car-sharing service (image: Rosenfeld Media)

Ecology maps are particularly useful for designing connected products because they show relationships among different *categories* of elements: people, regulations, infrastructures, applications, and so on. Those relationships could include: enrollment, family bonds, payment, legal liability, corporate governance... anything at all that could connect entities together. They represent business models along with technical requirements. Figure 6-9 shows a service ecology map that includes, among other things, “citizens,” “traffic,” “keys,” and “contract.” In theory, there’s no end to the details. At this point in product definition,

however, the best ecosystem maps are those that help you work out how design principles might play out in practice. A Business Origami workshop (described in Chapter 14) is a good way to start.<sup>15</sup>

Building a service ecology map can itself generate new ideas. Listing the actors and visualizing the relationships among them will probably trigger some brainstorming about how things could be better, and how to bring those changes about. With the important elements made visible, you can start hypothesizing about what would happen if you removed certain actors, swapped in new ones, or changed the relationships among existing ones.

Think about automation, often seen as the obvious next step for many tasks. Certainly, you can install an automated watering system to keep your plants from dying. Or you can use a wireless moisture sensor such as Botanicalls ([www.botanicalls.com](http://www.botanicalls.com)) that telephones or tweets when the plant needs more water. Both options will help plants survive. However, the first option means paying less attention to your plants and their environment. The second means spending more time on gardening. For the many people who find gardening relaxing and healthy, there's no question that the second option is preferable. Working with ecology maps can help us see past the obvious choices to potentially more interesting solutions.

Because service ecology maps often get very complex, it is helpful to use visual cues to focus attention on a few key relationships. Figure 6-9 uses color to highlight two types of elements: motivations for participation and enablers of use. Figure 6-8 foregrounds the activities of moving house, making the tools, actors, and stakeholders less prominent.

## Recurring Questions for Product Strategy

Over the years, we have noticed that certain business strategies, implemented with the best of intentions, can spur public disputes when *user experience* radically violates *user expectations*.

---

<sup>15</sup> Handy instructions for ecosystem mapping are available from Hussain, S. (3 February 2014). Designing Digital Strategies, Part I: Cartography. <http://bit.ly/1bDbVhn>.

Expectations, of course, change. They are neither universal nor eternal. However, consumer product design deals not with the far future but with the situation *right now*.

We can sum up many of the potential issues as imbalances in effort and reward. That is: *Who (or what) does the work necessary for the product or service to function? Who (or what) reaps the rewards? Are those rewards meaningful in light of the effort, or cost, of providing or using a service?*

In the previous chapter, we pointed out that a home security system might have to accommodate parents, small children, teenagers, pets, delivery people, and so on. If your home security product requires labor from people who don't find it inherently rewarding, don't feel like their effort will be compensated appropriately, and don't care about poor outcomes (e.g., teenagers), it is likely to fail. The time to start dealing with those imbalances is during product definition, because they have consequences for business models, user experience, and technical implementation.

Throughout this chapter, we've referenced three common sources of friction:

- Persuasion and behavior change
- Data ownership and data sharing
- Machine automation and human action

You may not need to consider every one of these as you define a new product, but it's worth keeping them in mind as questions for debate rather than taking their relevance, or irrelevance, for granted.

## **PERSUASION AND BEHAVIOR CHANGE**

Connected device makers are often in the business of behavior change. Activity trackers like the Fitbit aim to help people exercise more, Opower's data-driven reminders aim to reduce energy or gas consumption, and Target's "pregnancy coupons" aim to induce more spending. Networked pill bottles like the GlowCap promote medication compliance, and networked scales like those from Withings make weight management more collaborative. These business models depend on the collection of data about human behavior, in real time, and the predictive analytics to make inferences from them.

Changing habits takes a lot of work. The difficulty, as we discuss in Chapter 11, is that users must commit to using the product before they see results. If you're designing a product intended for behavior change, then any calculation of effort and reward needs to factor in not just the work of changing one's habits, but also the work of using and maintaining the product intended to foster that change. So who or what benefits from all that work?

If you buy an activity tracker or other self-monitoring service, then you are doing all this work for yourself. The actor doing the hard work—that's *you*—reaps the rewards...if it works. Designers can move past the initial value questions to designing *the thing right*, so that buyers keep on using the product past the initial flush of enthusiasm.

Different questions arise when the people doing the hard work of behavior change may not reap all or even most of the rewards. Users may not want to change their habits to suit buyers or product managers. How might you help negotiate that tension?

In some cases, tension among different stakeholders opens opportunities for product and service design. Remember, Opower lists one of their missions as *improving* the relationship of utilities and their customers. The company acknowledges upfront that the relationship could be improved. That's why we emphasize modeling the perspectives of multiple stakeholders, not only users. Value tensions or conflicts can productively motivate persuasive product design, but only if you first identify and acknowledge them. We talk more about identifying conflicts among needs and roles in Chapter 5.

## **DATA OWNERSHIP**

Designing for connected products is, in the words of Chapter 13, *designing with data*. By “data ownership,” we mean roughly, (1) what kind of information about users does your product collect and (2) who can access it? If you think that sensing or modeling human behavior is part of how your product works, then the product definition stage is a good time to start mapping out a potential range of reactions to it.

Take the Pay As You Drive program from Progressive, an insurance company. A custom device plugged into a car's onboard diagnostics port tracks driving habits for six months, after which the driver's insurance may be discounted to reflect what Progressive considers “good driving.” The device does not track or report location, and the data is

only released in case of accident. Some observers saw it as a useful way to pay less money for coverage. Others saw it as invasive surveillance.<sup>16</sup> As one industry observer wrote:

Are you on the side of giving up some anonymity in order to save, or would you like to keep government and large companies out of your personal behavior?<sup>17</sup>

The proliferating legal and governmental debates over corporate tracking of online behavior, “the right to be forgotten,”<sup>18</sup> the repeated uproar over various data storage services not being as secure as promised—all of those speak to a pervasive anxiety about what strangers know about our “personal behavior.” These anxieties are not so much about the letter of the law—everyone agrees, for example, that Pay As You Drive is entirely legal—but whether corporations *ought to collect it*.

And as with the Pay As You Drive program, questions about data use are often summed up as questions about *privacy*—that is, what other entities should be able to learn about individuals. However, definitions of “privacy” are highly variable. Different cultural groups may have very different expectations for what sort of information is sensitive, who has the right to access it, and under what circumstances it can be shared. Moreover, expectations can change over time as people become habituated to new types of services. Before Facebook, who ever expected to have our morning news include selfies of casual acquaintances violating state and national laws? Not us, that’s for sure.

Here are some questions to ask yourself as you start thinking about how to fund your project:

- Are any of your users likely to call the information you’re collecting (or the inferences you draw from it) “personal” or “sensitive”? What about the legal systems that govern conduct—whether your conduct or your users’?
- With whom are you planning to share that information, or those inferences?

---

16 “How’s My Driving?” *The Economist*, February 23, 2013, <http://econ.st/1bDc84c>.

17 K. Gordon, “Spying for Savings: Progressive Snapshot,” <http://bit.ly/1bDcdFo>.

18 See *The Guardian*’s online archive of “right to be forgotten” articles for a helpful primer on the term, especially as used in Europe and Britain: <http://www.theguardian.com/technology/right-to-be-forgotten>.



- Are any of your users likely to object to sharing that information or making those inferences available to others? What about the relevant legal authorities?

If the answer to any of those questions is “yes,” arguing that your service falls within the letter of the law will probably not pacify opponents. If the answer to any of those questions is “I don’t know,” you should probably do some research. If you’re in an organization committed to treating data from users in ways that would upset them, you should at least plan how you’ll respond when what you’ve been doing becomes public.

For us, the first commandment of good design is *Do your best not to violate your users’ values*. New products and services may well cross the line for some people, as with the automotive journalist we just quoted. The wise company, like Progressive, will anticipate those objections and develop strategies for mitigating them. For example, Progressive doesn’t track location and removes the devices after the trial is over. Progressive even calls the device the “Snapshot” to communicate that it is more like an informal photo (there’s another metaphor!), not a continuing data stream. But in order to take effective steps, you need to stay sensitive to those potential objections.

We’ll discuss the ethical questions of connected products further in Chapter 11, alongside the technical implications for security, privacy, transparency, and the right to control one’s data.

## **AUTOMATION**

There’s a tendency in connected products and services design to assume that having machines take over human work is a good in itself. Who doesn’t want to offload work—whether mental or physical—onto a machine? However, there are good reasons to treat automation as a choice, not a default. Instead of talking about automation, philosophers of technology often talk about “delegation”—that is, the distribution of responsibility and effort among different kinds of humans and machines.<sup>19</sup>

---

19 A reasonably accessible introduction is: B. Latour, “Where Are the Missing Masses? The Sociology of a Few Mundane Artefacts,” in *Shaping Technology-Building Society: Studies in Sociotechnical Change*, eds. W. E. Bijker and J. Law (Cambridge, MA: MIT Press), 225–259. Freely available from <http://www.bruno-latour.fr/node/258>.

Consider a home security service that helps neighbors look out for each other instead of *delegating* the responsibility to security cameras and remotely operated locks. In that example, having either a human or a camera keep a watch on the front door produces very different services, very different value propositions, and very different models for what the system is and how it works. And it's not clear that one is necessarily better than the other. Many resource-sharing services (e.g., Lyft, Airbnb, etc.) are popular precisely *because* they automate one part of the service while creating more work for humans elsewhere. Or consider our gardening example—pruning one's roses might be labor intensive, but it is also part of what makes gardening worthwhile.

## Summary

*Are you making the right thing?*

Our goal is to use our understanding of users and other stakeholders to define products that people are more likely to use and like using. But it's a complex process. Users and other nonexpert stakeholders' world-views and values often conflict with those instantiated in typical technologies and business models. And often, they don't have the technical expertise to predict long-term consequences of use.

If users don't immediately feel a benefit in proportion to the costs of using the product, no amount of interface wizardry will solve the problem. Moreover, discovering the long-term costs of common business models can anger previously unaware users. We touched on persuasion, data control, and automation as frequent sources of tension between business models and user expectations. We've also introduced some tools for identifying and anticipating those tensions.

We strongly recommend negotiating, not ignoring, these conflicts. We're not demanding that you resolve every tension immediately. Nor are we telling you to *never ever* surprise your users or do something they might not like. What we are saying is that product definition is the time to begin identifying and confronting potential mismatches in benefits and costs across stakeholders.

Before making glossy renderings or detailed product plans, make a succinct "elevator pitch" product definition in the form of:

*Product name is a(n) type of object that helps category of people name of job to be done by technical means.*

The elevator pitch sums up what the product is, what it does, and why it's important.

To identify the *job to be done*, you ask what actors and stakeholders value—and how a product could help them keep it or gain it. “Value” doesn't just mean tangible benefits (like saving time or money). It also includes personal aspirations and definitions of moral behavior. *Portraits*, *immersion tools*, and *journey maps* are three techniques for drawing out values from research data and communicating them.

What *type of object* are you making? Using a metaphor for the product's conceptual model can help people relate new and unfamiliar technologies to what they already know. However, careless metaphors cause trouble when they assume knowledge the audience doesn't possess, or don't resonate with an audience's existing worldview. The *product name*, such as “Snapshot,” “FuelBand,” or “Mother,” is also a way to bring a metaphor into the elevator pitch.

Through what *technical means* does the product work? Design principles are concise, product-specific imperatives. By guiding concept generation and evaluation, they translate values and metaphors into decision making. A *service ecology (or ecosystem) map* helps decide where and how to apply those principles. It combines stakeholders, their attributes, and contextual factors into a simplified diagram showing connections among the most important elements.

The examples in this chapter illustrate three sites of frequent imbalances in the benefits and costs perceived by various actors and stakeholders: persuasion and behavior change; data ownership and sharing; and machine automation and human action. Translating engagement and research directly into product definitions is the best way we know to anticipate potential imbalances, and then to either mitigate tensions or turn them into advantages.

The next chapters will start to address Buxton's second challenge: *how to make the thing right*.



## [Case Study 2]

# Little Kelham: Connected Home

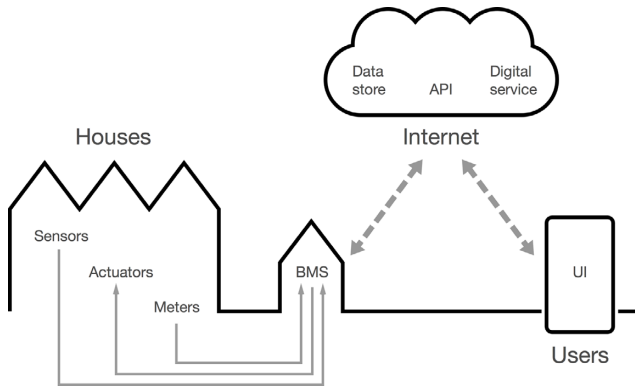
**BY MATT EDGAR**

Little Kelham in Sheffield, Northern England, is a development of 150 houses by sustainable property developer Citu. Each home is insulated and ventilated to a “passive house” standard that needs far less energy than a conventional home to stay comfortably warm all year round. To complement this low carbon living proposition, Citu wanted a digital service for homeowners to monitor and control their energy and water use.

The development has a strong community dimension, too. Instead of separate solar power from the roof of each house, residents will share the benefits of electricity generated right across the site. When they do use energy from the grid, they’ll buy in bulk as a community. So the digital service we designed for Little Kelham had to be true to this vision—not just something for individual homeowners, but a service that would help the community as a whole collaborate to make the most of their resources.

To make it happen, we had to first get to know a full stack of IoT technology, working across the site as a whole, and be sure all the parts would play nicely together to deliver a good experience to every user (see Figure CS2-1):

- Each house contains “edge devices”—the sensors, actuators, and meters
- There’s a “gateway device” in the form of a building management system (BMS) for the whole site
- Data is stored and the BMS controlled by the digital service through an Internet API (application programming interface)
- The digital service has a responsive web user interface (UI) for homeowners to access at home and remotely



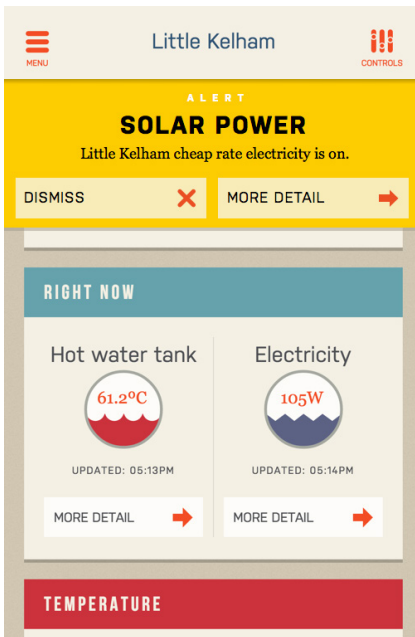
**FIGURE CS2-1**  
The energy system at Little Kelham

The full user experience runs over the Internet, but the BMS and in-home hardware can still operate and back up data locally if they ever lose their connection.

Little Kelham’s distinctive proposition and context drove hardware choices from the start. The UK government’s smart meter standard prescribes readings every 30 minutes. This would risk (in northern England at least!) missing the chance to use solar power generated during a brief sunny spell. Therefore, meters at Little Kelham needed to report much more frequently—once a minute instead of once per half hour.

Our UX design also responded to the profile of Little Kelham buyers, and behavioral insights from another sustainable development, Greenhouse in nearby Leeds. The majority of buyers were professionals who would be out at work during the daytime when solar peaks were most likely. We didn’t want them to miss out on cheap electricity, so we built in an option to automatically heat a well-insulated hot water tank whenever solar power was plentiful (see Figure CS2-2).

In construction, some things are literally set in concrete, but IoT services can and should be more malleable than that. Knowing requirements would change over time, we took a user-centered, agile approach to the design process. We focused the first release on helping Little Kelham buyers get to know their homes through a few simple visualizations and controls. More sophisticated functionality could be left for future iterations to be codesigned with residents according to their emerging needs.



**FIGURE CS2-2**

Alerts let users know they can take advantage of cheap electricity generated from on-site solar panels—this is a key part of the Little Kelham proposition that the UX needs to support

At-home research visits helped us build empathy with people who were moving house. Everybody working on the digital service went on at least one visit to someone who was in the process of moving, or had just done so. In kitchens surrounded by boxes and bubble wrap, it was easy to see how the first few stressful weeks after moving in are vital in creating patterns of behavior in the new home. Unfortunately, the way utilities have traditionally worked, it can take months for someone moving into a new home to get a believable electricity, gas, or water bill. By then, they've most likely settled into ways of living there—no matter how inefficient—that will be difficult to shift. Making energy use and costs much more visible in the first weeks of living somewhere emerged as a key opportunity for our new service.

Utility usage and control are often presented in a mechanistic way, with graphs and skeuomorphic representations of transistor-age dials and controls. Yet listening to residents it was clear that there was an opportunity for a much richer palette of interactions. Interviews with

existing users of smart energy monitoring revealed ways in which the need for data could vary over time. For example, one Greenhouse resident recalled: “In the first year I used to check it a lot. Now I’d probably check it more when it starts to get warmer or starts to get colder.”

Thinking about this initial learning period was especially important for buyers at Little Kelham. They would be moving from conventional houses to “passive” homes. Rather than making “smart homes” that would try to second-guess their inhabitants’ future intentions with complex theoretical rules, the aim was to make “smart homeowners” who would feel more confident and in control of their new living environments.

To broaden the palette of interaction in heating control, we referred to the three levels of emotional design described by Don Norman: visceral, behavioral, and reflective.<sup>1</sup> Standard heating controls—even the few well-designed ones on the market—tended to focus on the behavioral part, the pleasure and effectiveness of using the controls. By adding usage monitoring and billing, Little Kelham’s digital service would support a more reflective response, the ability to rationalize and understand home energy management and utility costs on an intellectual level. Thinking of energy data as a social object to be discussed among members of the same household or with neighbors reinforced this further.

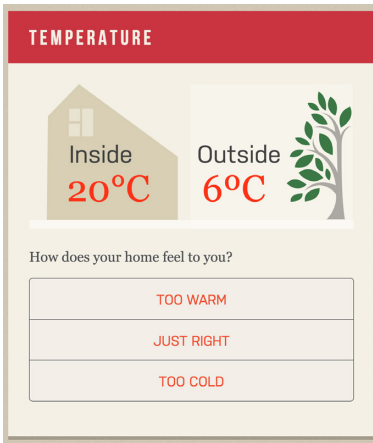
But a visceral reaction to the environment was also important: sometimes users just want to make a space feel warmer or cooler right now. In research visits, homeowners had shown us how, when cold, they would reach for the thermostat to raise the temperature, by overriding the timer or increasing the thermostat set point. To break the link between feeling cold and turning up the thermostat, the digital service gained a new screen that asked the user how their home felt at that moment—too hot, just right, or too cold. This returned tailored suggestions such as closing windows, or putting on more clothes, as well as turning up the heat. It became known in the project as the Goldilocks interface (Figure CS2-3). Helping people learn what “just right” feels

---

1 <http://www.jnd.org/books/emotional-design-why-we-love-or-hate-everyday-things.html>



like to them would be a valuable outcome in its own right, and the data from it could be fed back to building managers to help with fine-tuning the passive house technology.

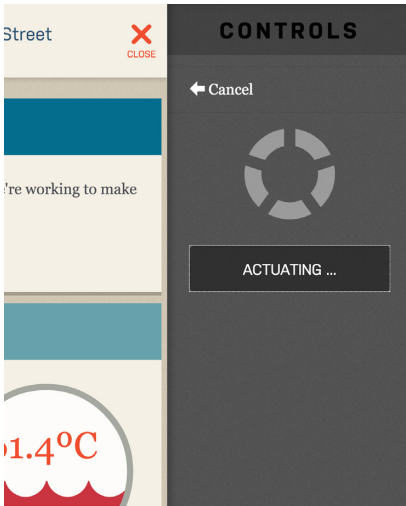


**FIGURE CS2-3**

Comfort is a highly personal and subjective concept; rather than expecting users to know what the “correct” temperature should be in degrees Celsius, the Goldilocks interface asks a question everyone should be able to answer without thinking—whether they are too warm, too cold, or just right

We tested an alpha version of the digital service with users to validate assumptions and refine the design before giving it to homeowners. This highlighted the importance of latency and feedback in the interface. Inevitably some equipment takes a few seconds to respond to remote control instructions and to return its new status. Showing users that something is happening during this delay helps to reassure and build credibility. Rather than building in irritating “are you sure?” dialogs around important actions, the service is designed to enable one-click actions, but with the possibility of undoing commands made in error (Figure CS2-4).

At the time of writing, the digital service is being deployed at Little Kelham. We’ll continue to support, learn, and further develop the service in partnership with residents as they move in and settle into their new homes (see Figure CS2-5).



**FIGURE CS2-4**

The service's responsive web app displays an "actuating" message with an animated spinner while it waits for confirmation that the user's command has been successful; as long as the spinner is displayed, the user can still cancel their command



**FIGURE CS2-5**

A CGI rendering of the development at Little Kelham

What's next? Well, more technically inclined residents should be able to download data about their homes, and even connect it to other tools and services using a secure, RESTful API. We also think a subset of home performance data could also become a valuable asset to pass on to new owners if the houses are sold.

# Embedded Device Design

**BY MARTIN CHARLIER**

This chapter is about the design of physical objects in the Internet of Things. In particular, the devices users interact with and that have embedded electronics. Compared to web or mobile UX design, thinking about physical objects requires an appreciation for some fundamental differences.

When reading or hearing about hardware startups, you'll often hear that "hardware is hard." This conclusion speaks to those fundamental differences to software. Designing software revolves around the ability to release early, iterate continuously, and easily address a global market. Hardware involves high upfront costs, can't change once manufactured, and every market requires individual due diligence.

In this chapter, we will look at some of the basics of designing physical objects, what's important to consider when designing physical *connected* objects, and how this is relevant to UX designers of an Internet of Things product.

This chapter introduces:

- Ways to think about physical object design with a focus on IoT (page 222)
- The differences between digital UX and making physical products (page 236)
- The industrial design process (page 241)
- The different facets of physical devices and their roles in creating great connected products (page 256)

This chapter addresses the following issues:

- Why physical device design is about more than usability and function (page 225)
- What UX designers need to know about the ideology of industrial designers (page 238)
- How concept generation, prototyping, and iteration work for physical devices (page 238)
- Tips and lessons for successful collaboration with industrial design teams (page 255)

## An Introduction to Thinking About Physical Objects in IoT

To begin, let's take one step back and first talk about the design of physical objects in general. In this section, we will touch on how the form of a product communicates, what different facets make up the design of a product, and where to start designing the physical devices in your system. This all frames the rest of the chapter and puts into perspective how this chapter talks about device design—that is to say, not just from a functional point of view.

### THE DEMISE OF “FORM FOLLOWS FUNCTION”

The form of physical products used to be heavily influenced by the technology within them. Television sets or old monitors had to accommodate a cathode ray tube, so they were big boxes that often tapered toward the back. Photographic cameras had to fit a film spinning from one cylinder into another with enough space to move it across a shutter, so they were cuboids.

This relationship between form and technology is one aspect of the well-known and somewhat tired principle of *form follows function*. It extends into the way we use products. Take the example of power tools. Masculine styling gimmicks aside, an ergonomic fit (and the technology inside) define the form. Controls are in easy reach of fingers and thumb. The form helps the function: aligning, switching on, and applying force with the tool.

None of this translates well into the world of digital products. Whether you're looking at an external hard drive, a router, a music player, a set-top box, a compact digital camera, or a smartphone, they all might as well be featureless black boxes of different sizes. As design critic Alice Rawsthorn points out: “[...] the appearance of most digital products bears no relation to what they do.”<sup>1</sup>

Picture for a moment an iPod shuffle (Figure 7-1). It's a thumb-sized device with a few buttons on one side, a clip mechanism on the other, and a headphone jack. If you weren't familiar with the concept of digital music files, the common iconography, or the size and appearance of a headphone jack, you had no chance of figuring out what this object does. As Rawsthorn points out, from this perspective, the most prominent feature is the clip mechanism. “Some sort of decorative clip,” you might think.



**FIGURE 7-1**  
iPod shuffle (image: Apple)

This is even more important when we enter the world of not just digital products, but the Internet of Things, where products connect to other devices to form systems that deliver services. That is because the kinds of devices that exist in the Internet of Things are more diverse and the roles they play more complex. Take, for example, the devices enabling many connected products in the first place: hubs, broadband modems, and WiFi routers. Explaining what this box (or sometimes array of boxes) does is another more abstract step compared to the iPod.

But why is this important?

---

<sup>1</sup> Alice Rawsthorn, “The Demise of ‘Form Follows Function,’” *The New York Times*, May 30, 2009, accessed March 25, 2015, <http://nyti.ms/1BuDf6K>.

Don Norman coined the term *affordance* for “the perceived and actual properties of [a] thing, primarily those fundamental properties that determine just how the thing could possibly be used.”<sup>2</sup>

Door handles are a simple example of this at work. Compare the two doors in Figure 7-2. A flat panel like on the door on the left “tells” you to push, while a handle like on the right “tells” you to pull.



**FIGURE 7-2**  
Different door handles (image: Martin Charlier)

Affordance is more than just understanding how a thing might be used. It is also about helping people make sense of it, to understand what it does or at least what sort of thing it is. For example, the six devices shown in Figure 7-3 cover products to do with heating, home automation, television, monitoring pets, and music.

Unlike in television sets or photographic cameras, today’s electronics rarely impose a particular form on the device. This means there are few archetypes or category conventions that make a device identifiable as having a certain function or purpose. So in many ways, the physical design of a connected object needs to work much harder than before.

---

2 Donald Norman, *The Design of Everyday Things* (New York: Basic Books, 1988), 9.

Sadly it seems that the opposite is often the case at the moment. More often than not connected products are nondescript plastic boxes with cables and blinking lights.



**FIGURE 7-3**

A collage of six different connected products: All are great products in their own right, yet it is hard (if not impossible) to make an emotional connection to the activities or contexts these devices relate to purely from seeing these devices

## **ANOTHER LAYER OF EXPERIENCE**

Beyond these practical, functional considerations, there is another layer of experience that comes into play when dealing with physical objects. Don Norman points out:

The virtual worlds of software are worlds of cognition: ideas and concepts presented without physical substance. Physical objects involve the world of emotion, where you experience things, whether the comfortable sensuousness of some surfaces or the grating, uncomfortable feel of others.<sup>3</sup>

---

<sup>3</sup> Donald Norman, *Emotional Design: Why We Love (or Hate) Everyday Things*. New York: Basic Books, 2004, 90.

Norman describes three levels of design: behavioral, visceral, and reflective:<sup>4</sup>

- Behavioral design is largely what we briefly described earlier: the aspects of usability and affordance, how the design makes us behave.
- Visceral design is less quantifiable. It is the kind of design that simply pleases our senses and is attractive. Similar to the way humans find symmetric faces more attractive, it is a sort of gut reaction. An example could be the hole pattern design on the Nest Protect smoke alarm (see Figure 7-4). The pattern used is unlikely to have a functional benefit over other pattern designs, but it simply makes the product visually enjoyable.



**FIGURE 7-4**

Nest Protect (image: Nest Labs)

- Reflective design concerns the image the design portrays and how it makes the owner feel. It can be about the statement it makes about the owner or the image it projects. Norman's example is a clever watch with an unusual but thought-provoking way of telling the time that he likes to wear on special occasions. You could think of this as “design to start a conversation” (even if with yourself). Rumor has it, that's what designer Philippe Starck once said about his famously useless Juicy Salif lemon squeezer (Figure 7-5).

---

<sup>4</sup> Ibid., 63–98.





**FIGURE 7-5**

Juicy Salif lemon squeezer—rumor has it designer Starck said about it: “It’s not meant to squeeze lemons, it is meant to start conversations.” (image: Alessi)

These levels of design play an important role in general, not just in physical object design. You can imagine how a digital user interface looks gorgeous, or how using a certain service is about making a social statement. Yet, when it comes to physical things that are on display in our homes, on our bodies, or in our cars, they are especially important. That is because these things are in the physical world. We have to deal with them, place them somewhere, see them, or get rid of them. Unlike software, we can’t close them, hide them in a folder, or minimize them. Whether in use or not, they take up space and we have to deal with them—even if just by seeing them.

Finally, there are many technical considerations that come into play, too. For example, if the device connects wirelessly, its antennas might require a physical form that has to be accommodated. The choice of materials can impact the performance of the antennas and the placement of the device can limit the options for powering it. All this can have knock-on effects to the UX of the system.

## INTERACTION AND PLACEMENT: BASIC DESIGN DRIVERS

Two primary considerations for the design of a connected device are the user interaction it involves and how conspicuous its typical placement is. If we analyze the relationship between these two considerations, three clusters emerge that a connected device will likely sit within (see Figure 7-6):

- Devices that are hidden away and only rarely interacted with beyond initial setup.
- Devices that are interacted with occasionally, but that are more conspicuous and abundant due to what they do.
- Devices that are interacted with frequently and that are likely to be on display to be easily accessible or visible.

Each of these groups needs a different design approach.

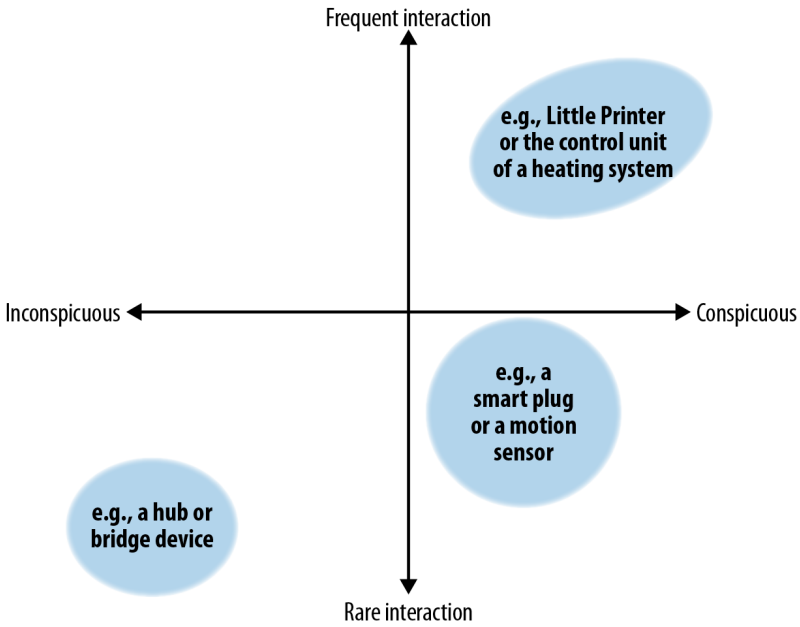
### **Devices that are hidden away and only rarely interacted with beyond initial setup**

*(Interaction: Rare; Placement: Inconspicuous)*

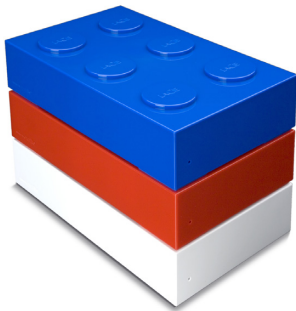
Bridges, gateways, and hubs are the physical devices required to make an IoT system work; they provide infrastructure. User interaction is often limited to the setup phase, after which the devices are almost forgotten about. Other than setup, further user interaction often only occurs in the event of issues. Users might want to press the “Reset” button or check which status LEDs light up.

The placement is often constrained by technical requirements like a wired connection to the main broadband router or a power connection. These devices are often hidden away in cupboards, corners of the room, or stuck behind shelves or other furniture.

The industrial design of devices in this group should embrace the interaction and visibility characteristics by making the device quite compact, stackable and placeable or mountable in different orientations. External hard drives have a similar design context and some manufacturers have embraced this by offering stackable designs (see Figure 7-7).



**FIGURE 7-6** Mapping the frequency of interaction and how “visible” a product is when installed; connected products fall under three common areas (illustration by Martin Charlier)



**FIGURE 7-7** Stackable external hard drive “Brick” by LaCie (image: LaCie)

The design might even take into account that the only interaction beyond the initial setup is likely to be a system reset. Why hide the reset button away, or make it difficult to operate if it’s the only interaction users are likely to do after setup?

An example from the Philips Hue connected lighting product line is the Hue Bridge (see Figure 7-8). Note how status LEDs and the link button are prominent central features of the device design. On its back are hardware features that allow it to be mounted on a wall and overall the device design is relatively ordinary.



**FIGURE 7-8**  
Philips Hue Bridge (image: Philips)

**Devices that are interacted with occasionally, but that are more conspicuous and abundant due to what they do**

*(Interaction: Rare; Placement: Relatively conspicuous)*

Smart plugs or motion sensors also provide a kind of infrastructure. But they are more specific to the purpose of the product and there might be lots of them. The placement, too, is more about the product purpose rather than purely connectivity. A motion sensor can't be hidden away like a hub, because most motion sensing technology requires the device to have a clear line of sight to the area it is observing.

Compared to hubs or bridges, users are more likely to move these devices around—for example, to rearrange smart plugs or motion sensors. And while interaction might not be on a daily basis, these devices are a little more top-of-mind than hubs or bridges.

This means the design should take branding and styling into account to greater degree. After all, because of the visibility, these objects will become a symbol for the brand or service. See the smart plug in Figure 7-9. Although it's a functional device, it features a soft and friendly shape as well as prominent branding.



**FIGURE 7-9**  
Belkin WeMo Switch (image: Belkin)

This also means that while devices might have practical design features for placement or arrangement similar to those we discussed in the previous group, these take aesthetics into account to a greater degree. For example, Tile, a Bluetooth tag that lets users find the products it is attached to, integrates a hole for attaching it to a key chain in more than a purely functional way (see Figure 7-10). The hole's size and placement is an integral part of the overall design of the product. It relates to the corner radii of the device and even features in the logo of the brand.



**FIGURE 7-10**  
Tile (image: Tile)

Looking at the Philips Hue product line, the light bulbs and to some degree the lamps and pendant lights fall in this category (see Figure 7-11). They aren't directly interacted with any more than the bridge, but by virtue of their function are placed much more visibly or on display. The difference in placement and visibility alone is reason enough for a different design approach: the Hue bulbs have more visible branding, a more refined industrial design using details like chamfers, convex and concave surfaces, and a more sophisticated use of colors and surface finishes. The various bulbs even share some design details like proportions and tapering toward the thread, suggesting that they are part of the same range of products.



**FIGURE 7-11**

Philips Hue bulb (image: Philips)

**Devices that are interacted with frequently and that are likely to be on display to be easily accessible or visible**

*(Interaction: Frequent; Placement: Conspicuous)*

The connected devices “where the action happens.” They can be only for user input, only for “output,” or a mixture of the two.

An example for an input-only device is the Amazon Dash (Figure 7-12), a candy bar–sized device that connects to its owners’ Amazon Fresh grocery delivery account via WiFi. It lets the user add items to their online shopping list by either scanning the barcode of an item, or by speaking into the device. Other than that, it has no function. Note that a device like this might still need a bridge to get connected, which would fall in one of the previous groups.



**FIGURE 7-12**

Amazon Dash (image: Amazon)

An example for an output-only device could be Voy Oslo’s connected Owl “Ugle” (Figure 7-13). It is remotely controlled to turn its head to a specific color patch on its body. The connected object has no other function or ability—it only outputs a value set elsewhere.



**FIGURE 7-13**

Voy Oslo Ugle (image: Einar Sneve Martinussen and Jørn Knutsen, Voy)

Of course there are also many in-between devices. A smart thermostat, for example, is a mixture of both input (set temperature) and output (review system status).

For this group of devices especially, the visceral and reflective aspects of design play a much more important role. That is because they are more openly on display and become symbols of the service. The design of these objects must deliver on more than just function. It also needs to live up to the less functional promises the product or service makes. So the most technically complex and important devices in a system might not be the most important when it comes to design.

In the Philips Hue product line, the Tap remote control sits in this cluster of products (Figure 7-14). Like the light bulbs, the device isn't likely to be hidden away like the bridge. Even though the basic form is similar to that of the bridge (a hockey puck–like cylinder), the industrial design is much more sophisticated. Surface details, colors, and material finishes are much closer to those visible on the bulbs than on the bridge. The device is also powered differently to make it moveable: it uses kinetic energy created when pressing its buttons to send commands.



**FIGURE 7-14**

Philips Hue Tap remote control (image: Philips)

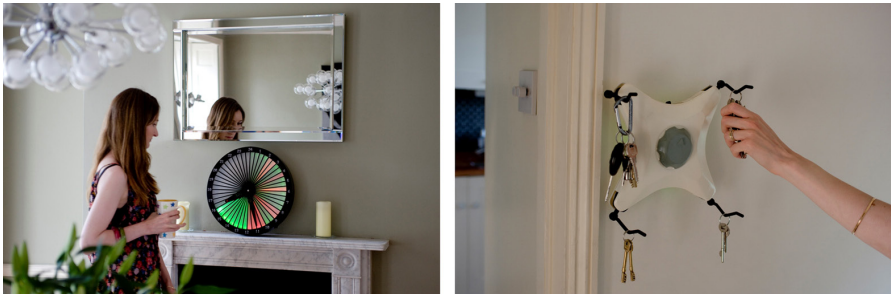
## **CHALLENGING THESE CHARACTERISTICS**

It's easy to jump to a conclusion about whether a device will be on display or not, or whether it needs interacting with or not. But it can be an inspiring exercise to challenge your assumptions in this regard. In



2009, Internet service provider TalkTalk approached a design course at Goldsmiths University in London and asked the course to design “future routers.”

The design students came up with concepts for routers that explored the prominence of the router as an object in the home and also its purpose (see Figure 7-15). One concept turns the router into a wall clock that displays the time of day combined with expected bandwidth. The concept acknowledges that bandwidth fluctuates and turns the router into an object that makes this information accessible. Another concept places the router near the entry of the house and includes key holders. The house can now use the presence or absence of keys to determine the presence of residents. When all keys are gone, the router switches itself and other devices off to save power (<http://bit.ly/1BuE9jz>).



**FIGURE 7-15**

Two of the router concepts created in the Future Routers project (images: Goldsmiths/TalkTalk)

Different design approaches are also visible in Google’s connected television products Nexus Q and Chromecast (see Figure 7-16). Both devices have a similar value proposition and involve a similar degree of interaction. But they show different approaches to the prominence of the device. While the Nexus Q is an elegant-looking sphere with sophisticated details, Chromecast is a small dongle that looks like a USB memory stick. The former has been designed to be on display while the latter disappears from view once connected to a television. Both devices alter the television experience, yet only one of them is easily visible doing so.



**FIGURE 7-16**

Google's connected television products, Nexus Q and Chromecast (images: Google)

At the time of writing, the Nexus Q is no longer available and Google appears to have replaced it with Chromecast. Perhaps this gives a clue as to which approach Google thinks is more appropriate for this kind of product.

## Making Stuff: Differences to UX

Now that we've discussed ways to think about connected objects conceptually, let's look at how the design process of physical objects differs to digital UX. The rest of this chapter is by no means meant to be a complete and exhaustive description. The aim is for UX designers to gain a basic understanding of industrial design. This section gives UX designers an overview of what basic knowledge they should have about industrial design.

### **PRODUCT DESIGN, INDUSTRIAL DESIGN, DESIGN ENGINEERING**

To begin, let's discuss some of the nuances that exist within the industrial design discipline. The exact titles designers and design agencies use vary and are also subject to a lot of debate in the design community. It's useful to be aware of the differences and able to probe and understand the profile of a designer or design team you might be working with.

Let's take three common titles: *Product Designer*, *Industrial Designer*, and *Design Engineer*. While there are no clear-cut definitions, there tend to be some differences between them. All are capable of designing physical objects and have an understanding of manufacturing techniques. But the titles signal different strengths and areas of focus.

Product Designers focus on a product's conceptual aspects—what it does, how it's used, how it looks—as well as give some thought to how it might be made. Industrial Designers, in contrast, are more involved in the engineering of a product and its manufacturing. Design Engineers have an understanding of the design process, but as the title suggests focus specifically on engineering and manufacturing. They concern themselves less with value propositions and the conceptual aspects. As mentioned previously, this is subject to a lot of discussions and many designers will happily debate and argue these semantics for hours.

For simplicity's sake, we'll use "Industrial Design" for the rest of this chapter.

The lesson here is that it is useful to understand that, like in digital design, there is a spectrum of skills that isn't covered by a single person, or single agency. Some might offer engineering services and manage manufacturing. Others might only develop concepts that engineering firms need to take further.

This is similar to the spectrum from design to development in digital products. Some designers get involved in technical decisions, others stay away from them. Some designers even work in code and could be described as designer-developers. A difference is that it is more common for industrial designers to have a good understanding of manufacturing than it is for digital designers to have knowledge of computer science.

Another difference common in digital design isn't easily found in physical object design: the split of UX and visual design as separate disciplines. A separation between designers that think about how a physical product is used and ones that think about how it looks doesn't quite exist. Industrial designers take a holistic approach that covers both usability and aesthetics. In digital design terms, one could describe industrial designers as constantly jumping between "visual design," "interaction design," "UX," and "development" in terms of their thinking.

## **MINDSET OF INDUSTRIAL DESIGNERS**

Another difference between UX design and industrial design is in the general mindset of the discipline. This has to do with their differing histories.

UX and interaction design originate from computer science, cognitive ergonomics, and the field of human–computer interaction. Industrial design has developed from architecture, the arts and crafts movement, and to some degree, engineering and theatre design. In parts, it is less scientific than UX. In contrast, UX increasingly involves analytics and measurements informing design decisions to optimize conversion or usability. Industrial design has remained less quantifiable and considers styling and aesthetics as much as it does ergonomics and usability.

This is evident in the way industrial design involves emotions and branding to a much greater degree. As an example, while UX design tends to concern itself specifically with the way a user completes a certain task, industrial design also concerns itself with considerations that UX designers might leave to visual designers or developers.

Another way of looking at these differences is the degree of codification of the discipline. UX design tends to revolve more around implementing “best practice” solutions. UX designers often use “pattern libraries” to sift through tried and tested solutions and find the most suitable ones to adapt for a particular challenge. This only exists in industrial design when it comes to details—say, a hinge mechanism for a battery cover. But on the level of the overall design, industrial designers work with a greater degree of freedom.

## **DESIGN FREEZE AS OPPOSED TO CONTINUOUS ITERATION**

The commitment involved in manufacturing is bigger than in software. For obvious reasons, you can’t make changes to a physical product once released, or offer an update for people to download. Modern software development with continuous iteration even after release means the “design” is never finished. In contrast, physical design projects have a “design freeze” point. This is the point on the project timeline when the current state of the design is “frozen in time” and handed over for manufacturing. After this point, further design changes would require postponing or even re-planning manufacturing. This fundamental difference means the design approach is different, too.

Because design decisions are not reversible once the product is manufactured, design teams project further into the future when generating concepts, do more upfront user research, and often prototype and test multiple alternative design directions at the same time. Once a direction is selected, the team goes on to develop it. In contrast, software design teams can start working directly toward a minimum viable product, release it, and then iterate and improve it. So rather than “just get started and then improve,” the approach of industrial design is to conceptualize the perfect product and then work toward an achievable version as close as possible to it.

This means multiple design directions are often explored in parallel for longer and developed quite far. That way options can be kept open and decisions made later on. In software, a single concept can be selected, developed, released, and completely flipped into something else for the next version based on feedback from releasing it.

All this doesn't mean iteration during the development is impossible. But with physical objects, it is often approached slightly different. When BERG developed Little Printer the team created functional prototypes that didn't look like the final product, but worked like it (see Figure 7-17). This allowed them to learn about how people would use and “live” with the device, and iterate the product experience. At the same time, the team iterated the appearance using a series of different models, iterating color and styling details (see Figure 7-18). Chapter 14 walks through prototyping in more depth, using another BERG project as an example. Although not always cost effective, additive manufacturing and electronics prototyping platforms increasingly make it possible to create small runs of devices that combine appearance and function.

These are ways to refine the value proposition and experience aspects of a physical product. But they don't uncover the problems and challenges of manufacturing. This can only be done through working with the factories that will make the device. Hence, design for manufacturability (DFM) is a design phase in its own right. This phase is all about achieving, tuning, and improving the actual manufacturing of the product. We'll talk a bit more about DFM later in this chapter.



**FIGURE 7-17**

BERG Little Printer experience prototypes (image: Timo Arnall)



**FIGURE 7-18**

BERG Little Printer nonfunctional appearance prototypes (image: Timo Arnall)

The kind of continuous iteration that is possible in software, even after a product has been launched, can be achieved within reason, too. For example, during the design phase of the digital service of sustainable property development Little Kelham (see the case study between Chapters 6 and 7), the team identified that users would benefit from a physical button placed near the front door of their homes. The initial hypothesis was that this button would be used to switch between “home” and “away” modes. What that meant exactly, however, was something the team didn’t have to decide finally. The function of the button wasn’t hardwired into the product and the team could iterate in software, even after the hardware was installed in homes.

Designing physical controls to be dynamic in such a way means the benefits from continuous iteration become accessible to physical products, not just software, within reason: designers still have to identify a requirement for certain dynamic controls and put them in place during the hardware design phase, something that wouldn’t be necessary in software.

## Essentials of the Design Process

Now that we’ve looked at some of the key differences between UX design and industrial design, let’s dive a bit deeper into the process of making physical products. In this section, we’ll look at some of the stages of the industrial design process that a UX designer is less likely to be familiar with.

Like in any user-centered design process, user insights, market research, and value proposition development are essential parts of designing physical things, too. But in the following, we will focus on the later project phases that are more specific to physical device design.

### **ESTABLISHING A DESIGN DIRECTION**

Digital UX design often operates within established platforms like the Web or Apple’s iOS. This provides a lot of constraints and building blocks right from the start.

This is different in industrial design. Hence the initial activities and tools in industrial design help narrow down the amount of choices, whether they be material options, shapes, colors, surface finishes, or stylistic details.

## Mood boards and design themes

Mood boards (also called “design themes” or “visual language boards”) are collages of pictures or words describing a conceptual direction or target audience taste. They can look at form factors, materials, colors, contexts, or other areas like architecture or fashion. Often, multiple mood boards are used to contrast and compare possible directions and provide designers with inspiration for their further exploration.

They are also used to describe and analyze competitors, design trends, or user profiles. Mood boards are a great addition to personas, a tool that should be familiar from UX design. In this case, the collages help describe what products, brands, CMF (color, material, finish), or environments are relevant to the persona. This is useful to set a design direction early and focus the concept generation phase. See Figure 7-19 for an example of three different design themes created for a recent project at insight, design, and innovation firm Evolve Collaborative.



**FIGURE 7-19**

An example of three design themes used for a travel product design project. Each theme provided an inspirational starting point for broad concept exploration. The themes are used to differentiate and organize the exploration while providing inspiration for form, detail, interaction, color, and materials (images: Evolve Collaborative, Portland, Oregon).

## CMF research

CMF stands for “color, material, finish.” In some industries (like automotive design) CMF is a discipline in its own right. CMF designers analyze color, material, and fashion trends, find new materials or finishing processes, and inform the design strategy.

In other cases (like in smaller design agencies), CMF is integrated into the design process and done by the industrial designers.



It often involves a lot of trends research and analysis. The longer time frames in industrial design mean a product might take a year or longer to be ready for market. CMF helps to ensure a product doesn't look dated before it is even released.

CMF research is about more than just looks. It's also about environmental considerations, like identifying recycled or sustainable materials for the product.

### **The importance of sketching**

Different kinds of sketches and hand drawings play an important role in the industrial design process. Throughout the project stages they fulfill various purposes.

After establishing a direction using mood boards, industrial designers often start exploring form ideas by quickly generating low-fidelity drawings. These “scribbles” or “thumbnail sketches” aren't doodles; they involve focus and concentration. They are used early on to generate a high quantity of ideas, forms, and directions, and document these ideas (whether they work well or not) so that the designers can move rapidly rather than laboring over single ideas. Sketching is both a visual thinking and a visual communication tool. In this early stage, it is all about exploring as many ideas and thoughts as possible in the shortest possible time, irrespective of quality or feasibility. This is a creative process where ideas are continually formed, documented, and taken forward as inspiration, which means even drawing mistakes or not obviously relevant input from looking at other products, fashion, or architecture provide relevant stimuli.

Using the different design themes or idea attributes like fundamental material choices, the quick scribbles can be clustered to help facilitate the decision making around what direction to explore further (see Figure 7-20).

As a design direction selected from such quick sketches is pursued further, sketches gain fidelity and detail. Industrial designers will start to draw various views describing the form in three dimensions and might also spend more time on selecting and showing colors, materials, and finishes using marker pens or computer software. Figure 7-21 shows such more refined sketches from a project at design and innovation consultancy Seymourpowell.



**FIGURE 7-20**

Quickly generated early sketches organized by relevant attributes like material type or the way features are integrated (image: Evolve Collaborative, Portland, Oregon)



**FIGURE 7-21**

Sketches exploring individual concepts from multiple angles and with more time spent working out design details (image: Seymourpowell)

As ideas and concepts are refined, combined, and built upon, the number of sketches decreases and the fidelity of sketches increases. Interpreting and “reading” early-stage sketches like the ones described so far tends to require a bit of an understanding of industrial design in general. Among designers, vague details that hint at a design direction are enough to understand what a particular sketch represents or embodies. That’s why when presenting and reviewing with nondesigners or designers from other backgrounds, industrial designers sometimes create even more refined presentation sketches using templates created in CAD and additional post processing using digital drawing tools.

However, ideas are often deliberately presented to clients using sketches, as opposed to entirely computer-generated photorealistic images. Sketches leave room for interpretation and are not understood to be too definitive. This means sketches can focus a conversation on particular areas. Like in digital design, using high-fidelity representations too early can lead to distraction when clients react to color or form details that aren’t decided yet. To manage this, it is not uncommon to make drawings “sketchy” for client meetings. Designers add “wrong” lines to make the sketches more ambiguous and prevent the conversation from getting bogged down in details.

Even at the late stages of projects, pen and paper are important. Construction drawings and technical sketches help ideate mechanical details and communicate with model makers or engineers. Pen drawings play an important role in the making of physical products. It’s important to understand how different types of drawings are used: to describe a general design direction, to propose a specific form factor, to describe a particular detail, or to document an exact specification of a part.

### **Formulating a design language**

“Design language” is about translating design strategy and brand values into different mediums. While “design principles” (discussed in the previous chapter) are concerned with interactions and behaviors, “design language” is about the aesthetic and visual manifestation of design strategy and brand values.

In the case of industrial design, this tends to encompass form, CMF, and design details like signature elements. The design language is an abstraction of the characteristics of a design and becomes a reference for other products from the same brand or product line. Once established, the design language becomes a guide for future industrial design projects.

A way of looking at it is to imagine a product or system as a person. How would you describe that person? How can you achieve this perception through industrial design? Is it sterile or playful? Is it clean or raw? Is it figurative or abstract? Organic or geometric? How do these qualities manifest themselves as shapes, colors, materials, surface finishes? Are there signature elements that are always integrated in a particular way?

A design language can help create coherence across products. But perhaps more importantly, it is a tool to define how the design embodies particular qualities and characteristics.

In a blog post about design language, creative director Luke Williams discusses this in more depth. He reverse-engineers the design language of early-generation iPods as an example. Why was the iPod's design so often described as "clean"? Williams links the industrial design to an environment that is all about "cleanliness": bathrooms. By referencing white porcelain and chrome, whether deliberate or not, the design supports this notion. (See Figure 7-22.)<sup>5</sup>

Design language becomes especially important in the Internet of Things. Connected products often involve a range of devices that are part of the same system. A common design language can help convey and reassure that the separate devices are part of one whole.

Design firm Seymourpowell worked with Korea Telecom to create such a common design language for their products (see Figure 7-23). Note how the diverse products are connected through consistent form details such as corner radii, convex shapes, and surface treatments.

---

5 Luke Williams, "The iPod and the Bathtub: How Products Shape Our Perceptions," Gizmodo, September 12, 2005, accessed March 25, 2015, <http://gizmodo.com/124912/frog-design-mind>.



**FIGURE 7-22**

iPod and bathtub (image: Frog Design)



**FIGURE 7-23**

Korea Telecom range of products unified by a design language (image: Seymourpowell)

For connected products, it isn't enough to create coherence across different devices. Design languages need to extend into helping convey the functionality, purpose, and role of the ever more devices and objects we will have. There are few conventions designers can build on. How can design language help differentiate a connected product from an unconnected one? This is a challenge that industrial designers will face when they work on connected versions of hitherto unconnected products. At the most basic level, this can mean the prominent

inclusion of established symbols to communicate connectedness. For example, the connected Crock-Pot slow cooker (see Figure 7-24) prominently features a common WiFi symbol on its main face.

## DETAILING AND DEVELOPING THE DESIGN

Once a few design directions have been selected, concept generation moves from sketches and drawings into three dimensions. Designers then create the data necessary to produce accurate mockups and photo-realistic visualizations of the product. But there is more that goes into taking a sketch of a device into 3D.

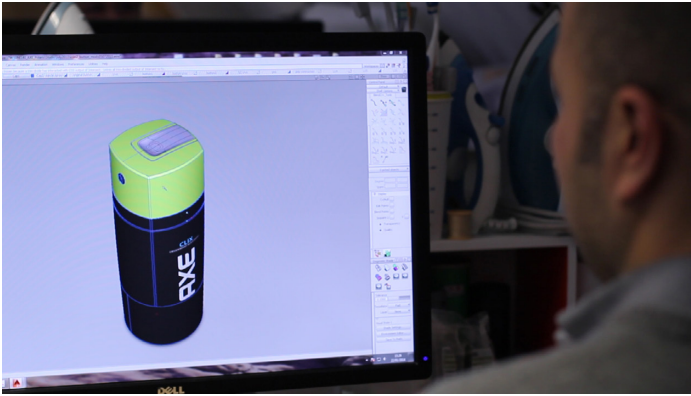


**FIGURE 7-24**

Crock-Pot Smart Slow Cooker with WeMo (image: Belkin)

### 3D modeling and rendering

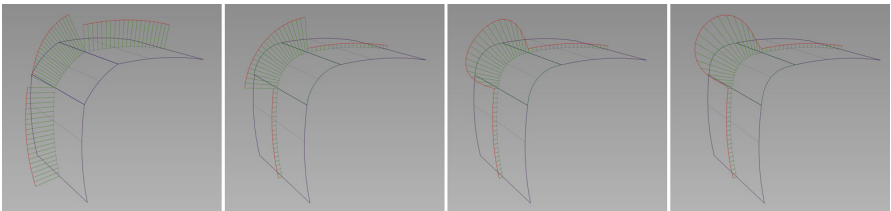
During 3D modeling, designers need to work out geometric details that they could ignore in a pen sketch. Surface geometry isn't always as straightforward as you might imagine and forms that might be easy to draw can be tricky to create in three dimensions. This is done using surface or solid CAD (computer-aided design) software (Figure 7-25).



**FIGURE 7-25**

A designer at Seymourpowell working on a 3D CAD model (image: Seymourpowell)

For example, when blending two surfaces into each other, industrial designers need to take extra care to define “surface continuity,” the smoothness of the transition. This is particularly important in surfaces that will be clearly visible in the final product, and when materials are used that are reflective (even just slightly). Designers use different terms to describe the degree of surface continuity (see Figure 7-26). This is about ensuring the tangents of each surface are equal at the transition. Higher degrees of continuity mean that even the curvature or the rate of change of the curvature is equal in the point of transition. This is important for creating not only a smooth transition, but making sure light reflections across this transition are smooth, too.



**FIGURE 7-26**

Different degrees of surface transition continuity made visible through “curvature combs”—from left: (1) positional continuity, note the gap in the curvature plot (red); (2) tangent continuity, curvature plot aligned, but not at same height; (3) G2 continuity, the curvature is aligned, but changes suddenly; (4) G3 continuity, the rate of change of curvature is smoother (illustration by Martin Charlier)

3D CAD models are also used to generate photorealistic renderings of the product, often using specific rendering tools such as Luxion Keyshot or Bunkspeed Shot. Tools like this are great to quickly simulate different color, material, and finishing options and present the product in different environments.

Sometimes, product renderings are even used to replace photography on packaging or advertisement. Believe it or not, 75% of the images in IKEA's catalog are renders instead of photography.<sup>6</sup> To IKEA, using computer-generated images makes creating and maintaining their catalog easier and more efficient.

## **CAD/CAM**

Computer-aided design and computer-aided manufacturing refers to software packages used to create 3D data and manufacture products.

Industrial designers often use a combination of software packages because there are trade-offs between them. So-called surface modeling packages let designers focus on creating complex freeform surfaces. But they're not ideal for producing data immediately usable for manufacturing or maintaining complex models that auto-update when engineering features are changed. In contrast, solid modeling tools are good at creating 3D data of high enough quality for manufacturing and tend to have parametric tools that keep a "history" of dimensions and features applied. This means updating an underlying parameter of a model (like a wall thickness) automatically updates an entire model accordingly. However, solid modeling packages often lack convenient features to create the kind of freeform detailing desired by industrial designers.

CAM software takes the data created in CAD further. Designers, model makers, or engineers use it to control computer numerical controlled (CNC) milling machines. It creates the tool paths a milling machine follows with its cutters to cut the model from a block of material.

---

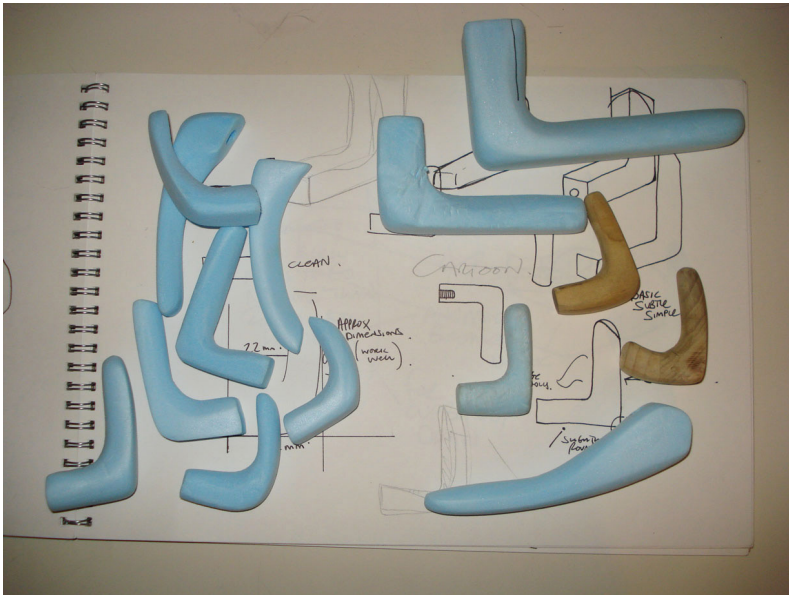
<sup>6</sup> Mark Wilson, "75% Of Ikea's Catalog Is Computer Generated Imagery," Fast Company, August 29, 2014, accessed March 25, 2015, <http://bit.ly/tBuEq67>.



## Model making and prototyping

Like with pen drawings, there are different levels of model making and prototyping that industrial designers use at different points in a project.

At early stages, designers use foam materials to rapidly create models by hand. Modeling foams made from polystyrene or polyurethane can be easily cut, sanded, drawn on, or CNC machined to generate physical mockups. These let designers explore the size and form of a device (see Figure 7-27). Paper and cardboard are used in the same way.



**FIGURE 7-27**

Handmade models exploring different forms for a door handle (image: Joseph Smith)

Such modeling foams come in different densities. Denser versions are harder to work with by hand, but more suitable for higher-fidelity models at later stages. Designers and model makers then use CAD and CNC machines to create models that can be painted to look almost indistinguishable from the desired final product. Models like this can help decide between, say, three different options for a design.

Aside from subtractive methods like CNC milling (where material is subtracted to create a model), additive methods like 3D printing are also used. Here a machine adds material only where necessary to create a model.

3D printing has enjoyed a lot of attention in recent years. But additive manufacturing (or “rapid prototyping”) has been a familiar prototyping process for designers for years. It can be used for early models of a product, but is more suitable for later stages when higher-fidelity models are required or mechanical engineering features need to be tested.

Such higher-fidelity models are also called “appearance models.” They represent a product concept, show the form factor, and might be spray painted to show the color or even surface textures. They can be 3D printed, but are often machined, because at least currently, CNC machines can achieve a higher degree of accuracy than 3D printers. In addition, the specific high-density foam materials are better suited for spray painting and finishing than the materials used in 3D printing.

## **ENGINEERING AND PRODUCTION**

When it comes to making product concepts real, design and engineering blend into each other. Making a product ready to manufacture is an enormous task in its own right. It requires engineering and manufacturing knowledge, as well as an understanding of how things are assembled in factories. A close collaboration with the factories involved is essential.

Two essential aspects of this stage that won’t be familiar to UX designers are design for manufacturability (DFM) and product certification. DFM is where the design is reviewed and developed further taking into account costs and efficiency of the actual manufacturing techniques. Product certification is about obtaining the required certificates that allow the product to be sold in different markets.

### **Hardware engineering, DFM, DFA**

Previously we talked about CAD modeling and creating the data to manufacture the forms of a product. In addition, a lot of engineering goes into areas not visible to most people. The knowledge required at this stage of product development means that designers often work alongside hardware or mechanical engineers. They might even hand over the project entirely.

Designers and engineers need to decide how to split parts to manufacture and assemble the device efficiently. Together, they design hinge or snap mechanisms. Or they place and design features to mount other components.

This phase sometimes reveals design changes necessary due to manufacturing methods or materials used. For example, designers and engineers need to think about where plastic is injected into a part and how it will flow through the mold. Even the rate at which the plastic cools down needs to be taken into account. Wall thicknesses or corners can cause problems and flat surfaces might warp due to irregular cooling of the plastic.

The care and skill applied during this phase decides whether people will perceive a product as low or high quality. What we pick out as elegant and simple products more often than not involves complex engineering and DFM.

In the documentary film *Objectified*, Apple designer Jonathan Ive describes the effort that goes into DFM. He points out that Apple's designers spend more time designing the fixtures and jigs used during manufacturing and finishing than the parts themselves. This shows how challenging it can be to achieve a beautiful and simple form factor. The real challenge might be in how a part is accurately held in place to machine or polish it.

Design for assembly (DFA) is another part of this area of consideration. The aim here is to take into account how a worker will put the product together, and how quality is assured. Again, the design of parts might need to change to allow for quicker assembly, or reduce the number of steps.

It's worth bearing in mind that in the area of connected products, not every product will need to be hardware engineered from scratch. For example, telecommunications service providers might not design and engineer their own router hardware, but buy and customize compatible off-the-shelf products that deliver the service. A designer of a connected product might be spending more time working with a supplier of hardware to customize and refine it, rather than develop and engineer it from the ground up.

## Product certification

Most electrical products need to be certified by authorities before they can be sold. In the European Economic Area, for example, this typically means having a CE (Conformité Européene) marking as well as a WEEE (Waste Electrical and Electronic Equipment) symbol. These mandatory marks declare that a product meets the requirements of the relevant European directives. In the case of CE, these directives concern, for example, the safety of low-voltage devices, electromagnetic compatibility, or compliance with RoHS 2, a directive restricting the use of certain hazardous substances in electrical and electronic equipment. A WEEE symbol is required by another directive and indicates to end users that the product needs to be disposed as electrical waste.

In the United States, a certification many connected products will require is the FCC Declaration of Conformity. This is a certification mark issued by the US Federal Communications Commission, a government agency that regulates electromagnetic interference.

With most of such certifications, specific requirements and tests are defined for different categories; some might be about safety and in other categories they might be about electromagnetic interference or use of certain materials. If a product falls within more than one category, it's necessary to pass a range of tests.

Some example tests are: Does the material used emit toxic gas when burned? Does it shatter into dangerous parts when dropped? Do the electronics emit electromagnetic signals that interfere with frequencies used by air traffic control or public radio stations?

Depending on the categories, certification can be carried out by manufacturers themselves or by authorized representatives. This is the case for some of the CE marking categories. Because the area of product certification is complex, specialized firms offer consultancy and execution of the required tests and documentation.

Product certification can have an impact on the industrial design of a product. For example, certain certification legislation requires additional security for the battery compartment if a product uses lithium

ion coin cell batteries. A recent change in legislation like this led BleepBleeps to delay their production to include an additional screw fixing in their Sammy Screamer motion alarm.<sup>7</sup>

## **COLLABORATION BETWEEN UX AND ID**

As the Internet of Things brings together digital and physical design, close collaboration between UX and industrial designers will become more commonplace. Being aware of differences, similarities, and the domain-specific tools improves these collaborations. Integrating design teams is a further step toward creating great connected products.

Design studio Clearleft explored this collaboration in an internal project. Through their internship program, they brought together an industrial designer, a web designer, and a technologist. Together, the team approached a broad briefing to “turn an active behavior into a passive one.” The multidisciplinary team jointly generated product ideas and design concepts.

The final concept was a modern-day jukebox named “Chune.” The product would get its music selection from the tastes and playlists of the people nearby (see Figure 7-28). The team brought the product idea to life by prototyping both the hardware and software aspects. They developed a functional physical mockup as well as UX prototypes for the software touchpoints.

The approach of one integrated team with different disciplines proved successful. Victor Johansson, the industrial designer of the group, described that the different backgrounds meant domain-specific ideas and knowledge came together into a holistic concept. Everybody informed every aspect of the concept and the team created a single design vision. No silos. Toward the later stages of the project, the team split up tasks according to their domain specialisms. Each team member could then work toward the joint vision in his or her respective area.

Projects like this are useful to explore how successful collaboration between software and hardware designers should be set up.

---

7 More about this in this update posting: <http://kck.st/1BuEu5G>.



**FIGURE 7-28**

Chune speaker prototype (image: Clearleft)

Jason Mesut, Head of Experience Design at Resonant Design and Innovation, brings together his industrial design background and deep experience in UX to help businesses integrate the two and develop experience-focused products and services. His recommendations match the characteristics of the Chune project. In order for UX designers and industrial designers to collaborate successfully, Mesut urges both sides to “unite under a common purpose,” “develop a shared vocabulary,” “share work regularly,” and “prototype together.”<sup>8</sup>

## Three Faces of a Physical Product

We’ve now covered an overview of industrial design methods and tools. Next, we will discuss the different aspects that make up a product, what they are about, and how they relate to connected devices.

We’ll break the design of connected objects into three areas:

### *Form, function, and usability*

What does it do? How is it used? What does the way it looks tell you?

---

<sup>8</sup> Jason Mesut discussed collaboration between UX and industrial design in a talk that is available here: <http://bit.ly/1BuEz9x>.

### *Aesthetics and appearance*

How does it look? What color does it have? What's the surface finish?

### *Materials, manufacturing, and maintenance*

What is it made of? How is it made? How is it assembled? How is it maintained?

## **FORM, FUNCTION, AND USABILITY**

Some of the first and most obvious considerations for the design of a connected device are practicality and ergonomics. What's the main purpose of the device? Who is using it? Do users need to hold or place it in a particular way? The design needs to make the product self-explanatory, easy to understand, and comfortable to use.

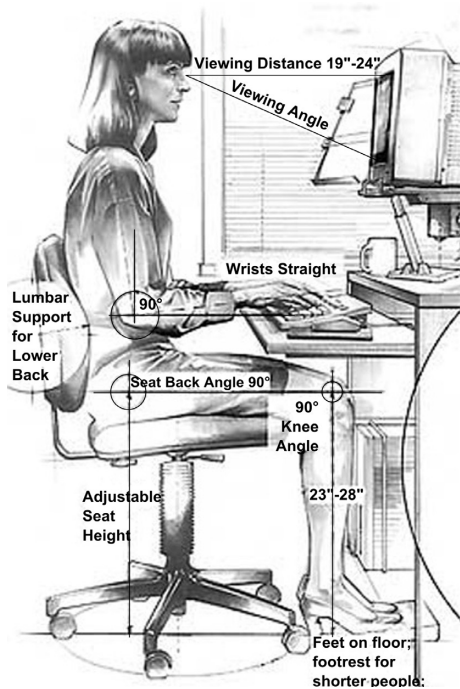
### **Anthropometrics and ergonomics**

The field of anthropometrics is about measuring the human body. It's about understanding dimensions, proportions, anatomy, and ability to move. The field of ergonomics (or human factors) uses this information to optimize the use of equipment and devices.

Take, for example, a workstation. The understanding of the human body informs the design to avoid operators having to work in uncomfortable postures. Figure 7-29 shows an illustration of measurements that the designer of the workstation would take into account.

In the design of connected devices this is important, too. For example, in the design of a smart plug, you might want to take into account the maximum grip strength of different grip types like power grip or pinch grip. The physical design should provide a good enough grip to easily remove it from a tight socket without causing the user pain.

Another aspect of ergonomics is to understand the physical context of use. For example, heating thermostats are normally mounted at chest height for reasons of thermal efficiency and the way the body perceives temperature. This means users might miss buttons on the top or bottom sides of the device, or find them uncomfortable to use.



**FIGURE 7-29**

An example of anthropometric measurements (image: public domain)

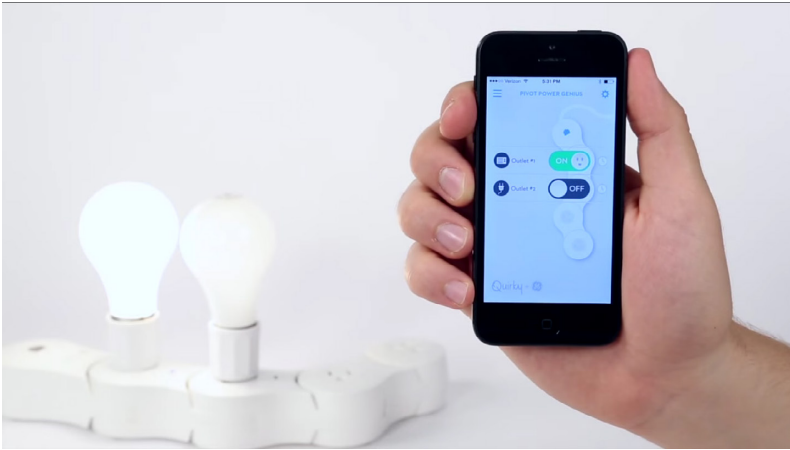
## Affordance

We've already briefly mentioned affordance and the classic example of door handles earlier. Another aspect of affordance is the use of mapping to help convey how something might be used or what it might do. An example is the buttons used to adjust the seat in certain car models. By making the button sizes and arrangement map to the parts of the car seat, users will find it easy to understand which button moves what.

In the case of connected objects, affordance is important to consider, too. The design of a motion sensor, for example, should give clues about what its field of vision is and how users should place it.

Mapping is another strategy to make a connected product or its user interface self-explanatory. An example is the Pivot Power Genius, an app-enabled power strip. An illustration of the strip inside the app maps to the design of the device. This clarifies which button controls what switch. (See Figure 7-30.)





**FIGURE 7-30**

Quirky+GE Pivot Power Genius (image: Quirky)

But using a very literal mapping isn't always advantageous and can lead to inappropriate UIs. It can be more important to follow platform conventions of a device. Chapter 9 discusses designing for consistency of UI elements and interactions across devices in more depth.

### **Practicalities**

There are also practical considerations that shape the experience a user has with a product, but that are often overlooked. Examples are battery compartments and how to open them, and whether users will need tools like screwdrivers to set up a device.

Other examples are bulky power adapters that render neighboring plugs on multiway sockets unusable or poor accessibility of ports to wire up the product.

If your aim is a great product experience, you should make sure such practicalities aren't de-prioritized.

### **Functionality and interfaces**

Another big aspect of the form and function of a device is of course how it is used. What functions does it have? What buttons do what? How is it operated?

Because this is such a big area in its own right, we will cover it in Chapter 8.

## AESTHETICS AND APPEARANCE

Industrial design goes far beyond function. We discussed earlier that it is a discipline that is as much about looks as it is about usability and function. This is less important for technical components that are hidden or integrated. But for things that take up space in our homes or things we wear on our bodies, aesthetics are important.

An article about the design of the Fitbit points out: “For a wearable device to be successful, it needs to do much more than just work: It also has to look good on you.” The designer Gadi Amit elaborates: “These objects are the most personable, nearly sensual or intimate objects. [...] They pose so many complex questions about your personality and fit to your specific human body. It’s a very delicate balance of emotional next to rational thinking.”<sup>9</sup>

Let’s look at the role appearance and aesthetics play in connected products.

### Consistency and cohesion: Making connections

Styling details and design language can be powerful tools to make sure a product is recognizably from a particular brand, or tie a collection of devices together conceptually.

For example, the devices from Estimote, a maker of Bluetooth beacons (see Figure 7-31) that can be attached to objects or locations, have a very distinct industrial design that uses shapes and colors also found in their app and company logo. The devices (as well as logo and app) use shapes similar to surface polygons used in 3D mesh modeling—where flat, usually triangular surfaces are used to describe a 3D model. In combination with a bold use of color, Estimote’s devices are easy to identify as being from the Estimote brand.

The aesthetic components of industrial design can also help conceptually connect multiple devices a user might have in their home. For example, the BERG Bridge, BERG’s Little Printer, and the remote control app share design details like color palette, fonts, and materials that link them together (see Figure 7-32). This makes it easier for a user to make the conceptual connection, too.

---

<sup>9</sup> Rachel Feltman, “The designer of the Fitbit says wearable devices need to be ‘nearly sensual,’” Quartz, January 20, 2014, accessed March 25, 2015, <http://bit.ly/1BuEJoy>.



**FIGURE 7-31**  
Estimize beacons (image: Estimize)

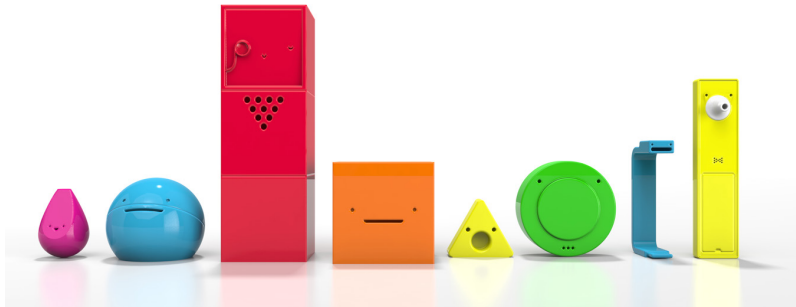


**FIGURE 7-32**  
Commonalities across BERG's Bridge, Little Printer, and app: use of color (light green and orange accent); typeface use in app and on bridge; material and finish of white plastic across bridge and the device itself (images: BERG)

In contrast, an off-the-shelf, or separately designed bridge device might not look related to the connected object it enables. This becomes more important when there are multiple bridges enabling different sets of connected devices. What relates to what?

Styling details and design language also tie a collection of things together under one brand or one value proposition. A motion sensor is a more or less generic sensor with many different applications. It can be hard for people to understand the value it offers or remember what system or service it is part of. Although currently just a concept, the BleepBleeps family of connected parenting devices (see Figure 7-33) uses appearance and styling to help with this problem. The product

range (only one of which has been funded through Kickstarter so far) shares a playful industrial design where each device is presented as a different character with an individual color, name, and (arguably) a face. The link of generic devices (such as a temperature sensor or a motion sensor) to the entire service is easier to spot and the function easier to remember given the specific theme of parenting and the witty device names (such as Tony Tempa, an ear thermometer; Sammy Screamer, a motion alarm; or David Camera, a baby monitor).



**FIGURE 7-33**

The BleepBleeps line up of concept products (image: Tom Evans—BleepBleeps)

Contrast this with the devices available for the SmartThings connected home platform (see Figure 7-34). The nondescript industrial design might help make the devices blend into their environment and be less conspicuous. However, the cues in the industrial design of each device (which help users remember and understand what each device does) are much more subtle, too.

### **Conveying brand image and brand promise**

The aesthetics and appearance of a product is also important to convey a particular brand image or promise. Imagine three connected toothbrushes: one with a minimalist form factor, referencing details from clinical equipment and a hospital color palette; one with toy-like details and a bright and varied color palette; and one made with what feels like cheap materials and a generic form factor.



**FIGURE 7-34**

A selection of the SmartThings products range (image: SmartThings)

How does each of these affect your expectation of the kind of service they connect to? Not just in terms of quality, but also in terms of what the service experience is about. Let's say the quality of data collection and analysis of each are the same. How much would you trust recommendations from the “cheap” toothbrush compared to the clinically sterile-looking one? What kind of an experience (say, “playful” or “serious”) would you expect from the toy-like one as opposed to the other two?

This power of styling is well illustrated in a project by designer Karin Ehrnberger. To explore gender-coded design, Ehrnberger swapped around the design languages of a power drill and a hand blender and made models (see Figure 7-35). The result is striking: the power tool seems weak and unreliable. The blender, on the other hand, seems dangerously overpowered for what it needs to do.

### **Personality and character**

Another way to approach the styling of a connected device is to give the object character and personality attributes. This can help convey functionality and other conceptual aspects of the product and wider system. An example is the design approach of BleepBleeps, which we mentioned earlier.



**FIGURE 7-35**

Karin Ehrnberger's mockups of drill and blender with swapped design language (images: Karin Ehrnberger)

BleepBleeps designed each device as a character with a name and color. Names relate to the device function. Sammy Screamer is a motion alarm, Tony Tempa a digital ear thermometer (see Figure 7-36.)



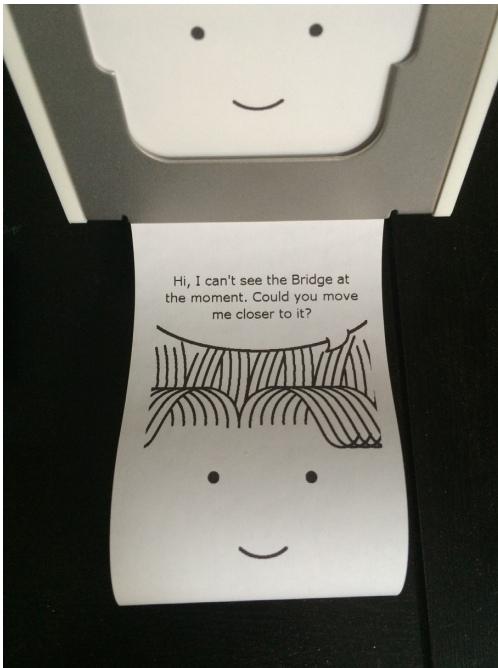
**FIGURE 7-36**

BleepBleeps Tony Tempa ear thermometer (image: Tom Evans—BleepBleeps)

The industrial design makes use of a psychological phenomenon called *pareidolia*, where humans see faces in objects or things where there aren't any. With only a little styling, a plastic device with a temperature sensor can become a curious-looking character using his/her/its nose to take a reading.<sup>10</sup>

All this makes recognizing and remembering which device does what much easier.

It can also make a product simply more enjoyable. BERG's Little Printer wouldn't be the same without its growing hairstyles and its happy or sad face. Besides adding enjoyable quirks, styling like this can make the interaction with a device feel less like "managing an appliance." Take Little Printer: when it loses connection, it prints a speech bubble asking for help (see Figure 7-37).



**FIGURE 7-37**

Little Printer's way of giving an "error" message (image: Martin Charlier)

---

10 Try for yourself here: <http://www.thingswithfaces.com/>.

### **Multisensory: Weight, texture, temperature**

The aesthetics and appearance of a product goes beyond visual qualities. Weight, materials, textures, and even surface temperatures play a major role in the experience of a product.

An example of this is the Beo4 TV remote control from Danish luxury electronics manufacturer Bang and Olufsen (see Figure 7-38). A typical remote, even for a high-priced television, weighs about 100g and is made of plastic. With 290g, Bang and Olufsen's remote is almost three times that weight. It uses metal elements that provide a cold sensation when touched. This isn't the case with plastic made to look like metal so it confirms the authenticity of the material. All this carries through the brand's image of high quality and luxury.



**FIGURE 7-38**

Bang and Olufsen Beo4 remote control (image: Bang and Olufsen)

Such sensual aspects of the device might need specific components. In consumer electronics, it's not uncommon to add weights simply for this purpose. When making mockups, designers can incorporate this by adding small blocks of metal into the model. That way a user research study can be sure to include the impression the weight of a product makes on the user.

Texture and materials can also include sensual qualities beyond color. ToyMail Mailmen, shown in Figure 7-39, are an example of this. They are connected voice mailboxes aimed at kids. The devices use a soft touch, almost rubber-like plastic. This provides protection, as it doesn't easily break. But it also feels friendly and nonserious compared to a hard, high-gloss finished shell.





**FIGURE 7-39**

Snort, one of the Toymail Mailmen (image: Toymail Co.)

In Chapter 8, we touch on functional aspects that are part of the aesthetic experience of a device, including how physical controls feel and what sounds a device makes.

## **MATERIALS, MANUFACTURING, AND MAINTENANCE**

The third area of consideration when designing a product is about the materials, manufacturing processes, and ongoing maintenance. In the following section, we will discuss some of the most important things to consider.

### **Operating environment and maintenance**

Unlike software, physical things collect dust, get touched, and are exposed to environmental conditions.

It's obvious that if you're designing a device that will be used underwater, it needs to be watertight. But there are less obvious environmental conditions that designers need to consider, too. For example, UV light degrades certain materials and surface treatments. We've all seen faded plastic on products that are a couple of years old. If a product is used outdoors, designers need to consider rain, cold, and heat. Even the mild acid coating of the human skin can alter a surface over time. These are just some examples that show how important it is to understand the operating environment of a device.

This is not just about protecting the device from its environment, but the other way around, too. If users wear a device with skin contact or come into contact with it frequently, designers need to ensure the

materials or adhesives used don't cause irritation or allergic reactions. Such allergies are not rare—for example, it is estimated that 10% to 20% of the US population is allergic to nickel, a metal found in many electronics.<sup>11</sup>

Cleaning a product is also an important consideration for the design. If the form has features that collect dust and are difficult to clean, it is likely to look dirty and unwelcoming after a few months in operation.

If users wear the device like an accessory or it's a piece of clothing, it may need to be washable. For example, the Mimo wearable baby monitor (Figure 7-40), which features sensors embedded into a baby kimono, is machine washable.



**FIGURE 7-40**

Mimo wearable baby monitor kimono (image: Rest Devices, Inc.)

### **Manufacturing constraints**

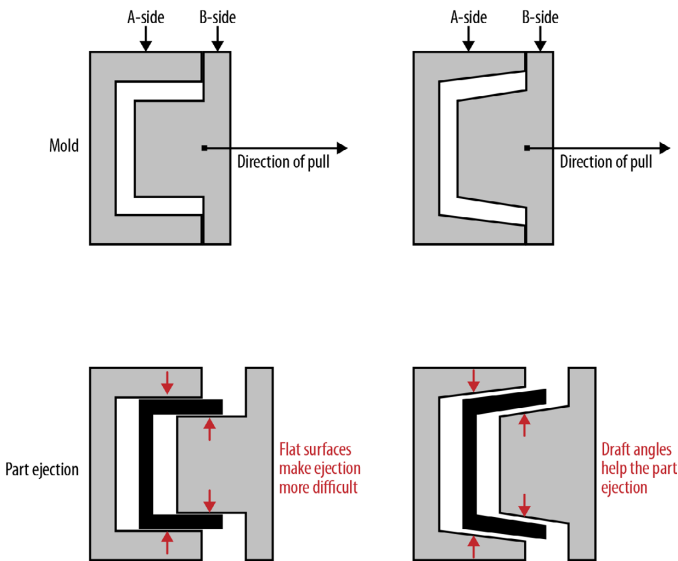
The way a product is manufactured provides a range of constraints. These include the form, color, texture, and finish of a product. Partly, this is to do with what is technically feasible, but it's also a cost consideration. Certain designs might be possible, but much more expensive to make.

---

<sup>11</sup> Rachel Abrams, "Nickel Allergies on Rise as Devices Meet Skin," *The New York Times*, August 20, 2014, accessed March 25, 2015, <http://nyti.ms/1BuET8q>.

Let's look at an example from injection molding, a process common in mass-manufactured devices. Hot liquid plastic is injected into a steel mold made of two or more parts, known as the "tool." These molds enclose a cavity defining the shape of the desired plastic part. The plastic flows through the mold, cools down, and hardens. Then the molding machine separates the tool so that the plastic part can be removed.

Even this simple and widely used process involves many design constraints. You can observe these in plastic products around you. For example, surfaces parallel to the direction of pull can be problematic. Parts with such surfaces are difficult to remove from the mold. You might have experienced two cups that fit perfectly into each other being difficult to separate. The friction involved in separating the part from the mold also puts strain onto the part, which can damage it. To avoid this, designers work with "draft angles." This means giving the surfaces in the direction of the pull a slight slant. See Figure 7-41.

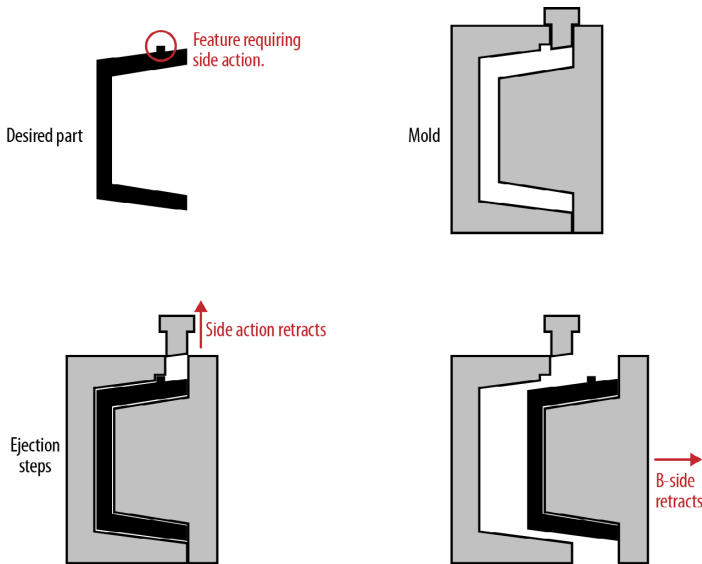


**FIGURE 7-41**

Illustration of a straight pull mold (left: no draft angles; right: surfaces with draft angles)

"Undercuts" are another example. Imagine you'd like a hole going through one side of your part. You'd need a pin-shaped feature in the mold that creates this hole during the injection process. But this feature is then in the way to remove the part from the mold. This is an

example of an “undercut,” a design feature that cuts under (or goes over) the mold pull direction. Creating undercuts isn’t impossible, but it makes the mold more expensive. A side-action would be required. The mold then requires three parts: the two main sides, as well as the pin that can retract from the cavity to release the part (see illustration in Figure 7-42). Designers aim to avoid side actions. They increase the complexity of the mold design, which increases costs.<sup>12</sup>



**FIGURE 7-42**  
Illustration of a side-action mold

In contrast, 3D printing does not involve a mold, and so “undercuts” are not an issue. Other design constraints exist though. Depending on the 3D printing methods used, designers need to consider construction geometry or escape holes. Construction geometry is additional material that is printed around the desired part. It helps support the part until it is finished and can support itself. This excess material needs to be broken away, cleaned, or even dissolved using acid. Designers need to consider how the operators can reach the material to remove it.

<sup>12</sup> See <http://www.protolabs.co.uk/resources/molding-design-guidelines/> for more examples of design considerations for injection molding.

Escape holes are openings in the part through which the operators can remove unused material. Some 3D printing methods work by selectively hardening areas in layers of material. This means unhardened (and reusable) material might be enclosed inside the part. Designers need to consider how this unused material can be removed.

These are just some examples of design constraints that the manufacturing process might impose on the designer. You can see how injection molding and 3D printing create entirely different kinds of design constraints.

Compared to injection molding, the constraints of 3D printing are negligible. But although it's a great method for prototyping, it's not currently a process for mass manufacturing. This means it can be deceptive. A design that is possible to create using 3D printing might be costly or impossible in a mass-manufacturing process. So, during the design, it is important to decide on the final method as soon as possible.

There are many different manufacturing methods, materials, and finishing processes. For designers interested in going into more depth, some useful books are:

- *Making It: Manufacturing Techniques for Product Design* by Chris Lefteri (Laurence King Publishing)
- *Materials and Design* by Michael Ashby (Butterworth-Heinemann)
- *Manufacturing Processes for Design Professionals* by Rob Thompson (Thames & Hudson)
- *From Concept to Consumer* by Phil Baker (FT Press)

### **Sustainability and recycling**

Environmental and ethical considerations also play a role in the design of physical products. Concepts such as the “circular economy” and frameworks such as “Cradle to Cradle” advocate a sustainable and waste-free product lifecycle. As consumers become more interested in the environmental impact of products, manufacturers become more open about it, too.

The Fairphone, a smartphone released in 2013, has such considerations at its center. The device uses only materials that are “conflict free.” This means their use doesn't fund armed forces or violent conflicts.

Material libraries such as Material Connexion offer specific services to find and identify such sustainable and ethical materials.

See Chapter 11 for more on the environmental impact of an IoT product.

### **BOM cost versus future-proofing**

When making a product, an important document to work with throughout a project is the *bill of materials* (*BOM*). It is the list of all components of a single unit with their respective costs. Traditionally, together with development and marketing costs, this would guide the retail price. Increasingly, other business models mean that devices are sold at or even below cost.

The role of the BOM is changing from the key document defining product cost to one aspect of a cost structure. It is still an important document for designers to work with. Until now, designers and engineers often aimed to reduce *BOM cost* as much as possible. In the case of connected products, this isn't necessarily good. Removing components or replacing them with alternatives to reduce costs can close the door on future functionalities. Designers need to make trade-offs between cost and longevity of a device. If your device includes components that aren't currently used, but that might enable future functionality, the additional cost of the device might be hard to justify for consumers comparing price and functionality at the point of purchase.

Designers of connected products need to strike the right balance. And traditionalist designers and engineers accustomed to a culture of reducing BOM cost need to change their way of thinking. They need to consider the BOM in the context of a roadmap and ecosystem that means components that aren't used right now are still important.

## **Summary**

The Internet of Things means that hardware and software design practices blend into each other. Connected products can neither be designed solely by industrial designers, nor by digital UX designers. Designing great connected products requires deep collaboration of these disciplines. This chapter provides a step toward enabling this kind of collaboration. Both sides have a lot to learn *about* each other to better understand and work with each other. Both sides also have a lot to learn *from* each other to push their respective practices even further.

Most importantly, as consumption moves back from digital to physical, all designers involved need to act responsibly. Unlike the apps that were hip for a while, and then weren't anymore, physical products don't

disappear easily. When they're not in our homes anymore, chances are they end up in piles of e-waste, often in poor nations, where they are being burned and dismantled to salvage the components and metals that can still be sold. In the process, toxic materials not only contaminate the environment but also poison the workers that scrap them.

Designing connected products is a new kind of discipline that combines digital UX, industrial design, and more. The designers of this new discipline mustn't forget the responsibility they carry toward our world's resources. And they mustn't forget that design is more than usability and efficiency—it is also beauty, craft, and art.

In summary, UX designers and product managers should be aware of the following characteristics of industrial design:

- Representing function through physical form and appearance is much harder for connected objects because there are few established device archetypes or category conventions, and many functions do not manifest in a physical form.
- In addition to functional requirements, the design of physical devices needs to consider emotional, aesthetic, and symbolic representative aspects.
- Frequency of interaction, how conspicuous a device will be once installed, and how many of a device users will have are basic drivers for the physical design. Although there are conventions for some device types about these drivers, it can be a worthwhile exercise to explore breaking with these conventions to arrive at innovative connected products.
- Like in software, designing devices requires a spectrum of skills rarely covered by one person or even one company. “Industrial design” is a term that includes many experts from aesthetics and styling to engineering and manufacturing.
- Industrial design has a different mindset and approach to UX. It is less task driven. It involves equal parts thinking about appearance, usability, engineering, and manufacturing.
- Both software and industrial design involve prototyping and iteration cycles. But industrial designers go through multiple cycles rapidly before releasing a product, while software teams can release earlier and continue these cycles to improve and update the product after the release.

- Industrial design is not bound to platforms that provide constraints and building blocks. Industrial designers use specific tools and methods of which UX designers should be aware.
- Pen and paper are still some of the most important and most used tools in industrial design—at every stage of a project. In the same way that industrial designers need to understand different types of UX sketches and drawings, UX designers should know how to “read” the different kinds of industrial design sketches and scribbles.
- Despite recent popularity, 3D printing is only one of many prototyping methods and is rarely used for mass manufacturing. UX designers should be aware that different manufacturing methods impose different constraints on the device design.
- Designing and selling a physical product involves challenges unfamiliar from developing and selling software, like manufacturing constraints or device certification. Physical appearance is only the tip of the iceberg.
- In order to build great experiences across all touchpoints, physical and digital, UX and industrial design teams need to be integrated and work jointly on an initial design vision.
- Decisions involved in the design of a product fall into three areas that are all interrelated: the way it is used, the way it looks, and the way it is made. UX designers should have a basic understanding of the kinds of considerations in each area.
- Aesthetics and appearance of a device play an important role in conveying brand values and in making products desirable and enjoyable. But they can also be used to improve the usability of a product.
- Real physical concerns like exposure to environmental conditions, material degradation, and what can and can’t be manufactured all influence the design of a product.
- Making physical products always also means making decisions on environmental, social, and ethical questions about material sources, manufacturing conditions, and the entire product lifecycle.



# Interface and Interaction Design

**BY MARTIN CHARLIER**

In this chapter, we will look at interface and interaction design for connected products. Many connected products involve web and smartphone interfaces. But there are many good resources available on these platforms already. This chapter will focus on the possibilities beyond point-and-click or touchscreen interfaces. This is for two reasons.

First, point-and-click and touchscreen interfaces dominate the majority of UX design work. But both interface types are less likely to be used on embedded devices like smart thermostats and or smart plugs.

Second, the Internet of Things provides an opportunity to redefine how we interact with digital devices. This point is brilliantly argued by designer Bret Victor in his “Brief Rant About the Future of Interaction Design.”<sup>1</sup> Victor discusses the diverse and intricate ways we use our hands on a daily basis. He critiques the many concept explorations that, although looking into the future, still revolve around touchscreens. The aim of this chapter is to inspire designers to experiment with new ways of interacting with products.

This chapter introduces:

- Different interface types and their benefits and drawbacks (page 276)
- Specific interface design challenges for embedded devices (page 312)
- Ways to design accessible and universal interfaces (page 320)

---

<sup>1</sup> “A Brief Rant On The Future Of Interaction Design,” Bret Victor, accessed March 25, 2015, <http://worrydream.com/ABriefRantOnTheFutureOfInteractionDesign/>.

This chapter addresses the following issues:

- Why having a screen isn't necessarily better than not having a screen (page 289)
- What strategic decisions designers face about the interactivity of a device (page 313)
- How interfaces can reduce the amount of attention asked from users (page 317)
- How devices can receive complex input through simple interfaces (page 318)

## Types of Interaction

Our bodies and abilities define how we can “talk” to devices (input). Our sensory system defines how we can “listen” to devices (output).

Yet, the way we interact with devices has so far been very limited. Tom Igoe and Dan O'Sullivan illustrated this in their book *Physical Computing*. Figure 8-1 shows their illustration of “how the computer sees us”: one eye instead of two because we only look at two-dimensional screens; one finger only, as we input through sequential tapping, which might as well be the same single finger; two small ears to hear stereo audio output from laptop speakers and the like. Igoe and O'Sullivan point out: “*To change how the computer reacts to us, we have to change how it sees us.*”<sup>2</sup>



**FIGURE 8-1**

“How the computer sees us” from *Physical Computing* (image: Tom Igoe, Dan O'Sullivan)

---

2 Dan O'Sullivan and Tom Igoe, *Physical Computing: Sensing and Controlling the Physical World with Computers*. Boston: Thomson Course Technology PTR, 2004, xix.

If the illustration were updated to include new interface technologies as they become available, it would gradually become more human-like. New interface technologies mean the computer is able to understand more about our actions and behaviors: seeing full body posture and gestures or listening to what we say.

This isn't to say new interface technologies necessarily improve the way we interact with products, but it shows how limited interfaces still are and how they only gradually enable new ways of interaction. This chapter aims to inspire you to think about different input and output possibilities. The following tables list some of the ways in which we could provide input, or receive output, and how they might be used:

INPUT THROUGH	USED IN...
Touch, Press	Physical controls, touchscreens
Movement and manipulation	Tangible UIs
Speech	Speech recognition
Whole body	Gesture recognition, proximity sensing
Galvanic skin response	Stress detection
Thoughts	Brain-computer interfaces
Heart rate	Determining stress, anxiety, sleep...

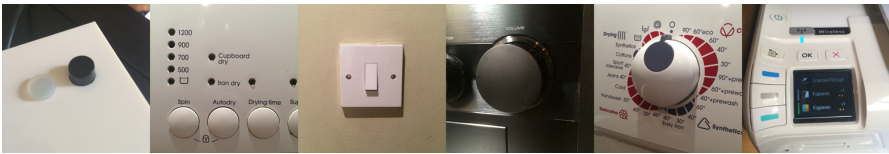
RECEIVE OUTPUT THROUGH	USED IN...
Seeing	LEDs, screens
Hearing	Sound, voice output
Tactile sensing	Vibration, force feedback, shape
Smell	Scent messaging
Temperature sensing	Temperature output

There are more interface types we could list. And not all the types we listed are readily applicable at the moment. Breaking down interaction into these components lets us combine them in different ways: a device might take voice input, but output only through a screen. Or it might have physical controls, but only respond using a voice through a speaker.

In this chapter, we will discuss combinations of inputs and outputs that are particularly relevant for connected products.

## PHYSICAL CONTROLS

Physical controls are everywhere: push buttons that activate something, switches that let us choose between two states, sliders and rotary knobs that let us set a value on a range or select between multiple settings. Figure 8-2 shows a few different kinds.



**FIGURE 8-2**

A few physical controls from around a house (images: Martin Charlier)

Physical controls aren't just input; sometimes they show a current setting. An electrical outlet switch controls the outlet, but also tells you if it's on or off. The tactile nature even allows you to feel the switch in the dark and use it without seeing it. In electronic devices, this can make things complicated. When users can change a setting from more than one place, controls might need to be motorized (like HiFi systems where the volume knob rotates even when the remote is used), or a combination of physical control and display is required.

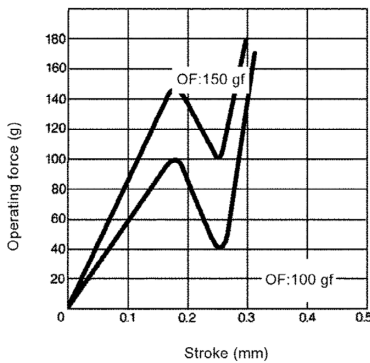
Physical controls are also expensive to change once your product is shipped. If you plan to update the functions of buttons later on, you might have to keep labeling abstract (through color coding or by using geometric shapes), or combine buttons with displays ("soft keys"). These options aren't always feasible and the cost of fixing design faults can be high. When Ford needed to fix a design fault relating to physical

controls, it required issuing a recall of over 13,000 cars.<sup>3</sup> Getting physical controls right is crucial, and button placement can make or break a product.

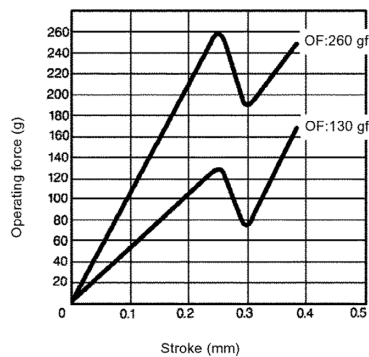
In addition to the placement, function, or type of control (button, switch, dial, etc.) designers also need to think about the tactile aspects. The haptic and tactile characteristics of a control influence the experience of a physical device. From the outside, they might be invisible.

To enable designers to work with the nuances that exist, the specifications for controls include operating force graphs. Figure 8-3 shows such graphs for different versions of what (from the outside) looks like the same button. Note the differences in both stroke (the distance the button travels when pressed) and operating force (the force at which the button “activates”).

**B3F-1** □□□/-3□□□



**B3F-4** □□□/-5□□□



**FIGURE 8-3**

Operating force specifications for different models of an identical looking button component (image: Omron component data sheet)

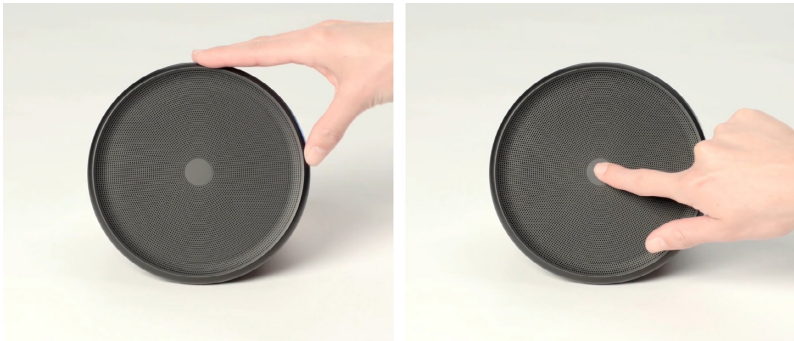
From the outside, a user won't be able to spot that difference. But it will make a noticeable difference to the way the interaction feels. This has a practical aspect: making sure a button is harder to press so users don't trigger it by accident. There is also an emotional aspect: making sure pressing a button feels satisfying rather than flimsy.

3 Chris Woodyard, “Ford recalls Lincoln MKC—twice in same day,” *USA Today*, January 4, 2015, accessed March 25, 2015, <http://usat.ly/1BuFYwK>.

To get an idea for how nuanced the tactile quality of a button can feel, just read a few reviews on the blog <http://knobfeel.tumblr.com>. It's not exactly a serious resource, but it gives an idea of how certain physical controls feel to somebody over other ones. For example, one of the reviews reads: *"I don't quite understand how it simultaneously is light to move, but [has] a heavy feel. The rotation can only be described as creamy."*<sup>4</sup>

Physical controls can become an identifying feature that makes a product stand out. Think of the click wheel control of the iPod. Another example is the Aether Cone (see Figure 8-4): a connected speaker that uses physical controls as part of its minimalistic interface. The device combines voice input with physical controls. It learns about personal preferences and automatically curates a playlist.

Rotating the front speaker grille slightly skips the current song. Rotating it further changes the entire musical genre. Pressing in the center of the speaker activates the voice input feature that lets users request specific songs. This interface is intriguing and makes the product stand out from its competition.



**FIGURE 8-4**

Some of Aether Cone's less obvious controls: rotating the speaker grille and pushing in the center of the speaker (images: Aether)

Using unconventional physical controls that are integrated with the device design can have drawbacks. Discovering the controls can be hard when they are nonobvious. Users need to learn where the controls are and how they work. Good usability can also be a challenge. Controls

---

4 More reviews at: <http://knobfeel.tumblr.com>.

defined by the form factor, instead of interaction requirements, might be suboptimal for the interactions required. Keeping controls to the bare minimum is often part of what makes a product simple and elegant. But too few controls can negatively influence the usability and value. A single control might need to do different things at different times. This means users have to switch modes (which can lead to mode errors) or accept less functionality. If you're using unconventional controls on your device, consider helping users with discovery and learning. To do this, you can utilize marketing and advertising to educate people about how they can use your product. The early TV adverts for Apple's iPhone took this approach. The short clips demonstrated how features worked by stepping viewers through the interaction. This has no doubt helped early customers know how to use the device before they even bought it.

Once a customer has purchased the product, the out-of-box experience (the moment where the product is taken out of its packaging) can help, too. Printed instructions that are revealed to users as they unbox the device or removable labels affixed to the product can convey essential information about its controls.

PHYSICAL CONTROLS	
<p><b>Are great...</b></p>	<p><b>... for direct and fast control.</b> Giving users access to something without having to find an app or navigate a menu. Think volume, mute, or camera buttons on smartphones to provide quick access anytime.</p> <p><b>... when fine adjustments are required.</b> Tactile qualities like friction and momentum can be leveraged. Controlling volume on a heavy, slow-to-rotate knob allows for finer adjustment than a slider on a touchscreen.</p> <p><b>... to make a product more accessible for users with impaired vision.</b> Blind users memorize button positions or affix tactile labels to guide them. Replacing all controls with a touchscreen makes the product far less accessible to this group (more on accessibility at the end of this chapter).</p>
<p><b>Are less appropriate...</b></p>	<p><b>... on products that change and develop through software updates.</b> For obvious reasons, it is hard to change controls around after the product is manufactured. Abstract labels (such as colors or shapes) or combinations with screens can keep physical controls dynamic.</p> <p><b>... when functions or settings can be controlled from multiple different places.</b> Controls might need separate displays to show state. They might even have to be motorized to move autonomously.</p>

## VISUAL AND SCREEN INTERFACES

Next, we'll discuss interfaces that communicate through visual means like using LEDs to communicate through light and the benefits and drawbacks of different screen technologies.

### Light output

Almost every electronic device has at least one LED somewhere showing that it is switched on or communicating a particular status. Connected devices often have more things to communicate. Hubs and bridges often have a number of LEDs to show a variety of connectivity statuses. When devices need to communicate even more, it can get complicated. Color-coding or blink patterns might be required to convey complex information through a simple LED. This can make understanding what is going on tedious. Table 8-1 gives you an idea of the sort of information that might have to be in the user manual.

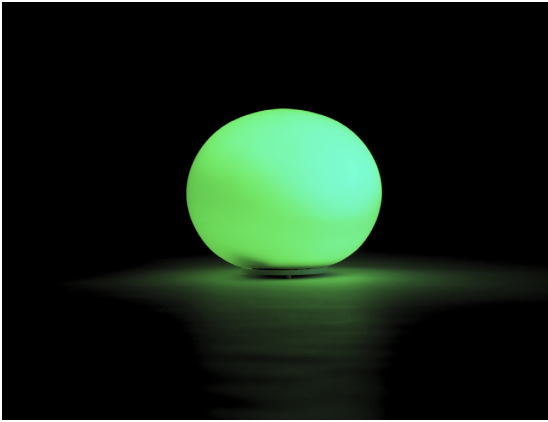
**TABLE 8-1.** An example of LED blink pattern and color information a connected device might need to provide in a user manual

Flashing red	Not connected
Flashing orange	Trying to connect but incorrect WiFi details
Flashing red, orange	Trying to connect but not succeeding (might be due to a firewall)
Flashing red, orange, green	Device booting up and connecting
Flashing green	Downloading firmware update
Solid green	Device connected
Not lit up	Device running (LED off to conserve power) or not powered up at all

Light can also be the main interface of a device. The Ambient Orb (see Figure 8-5) is a ball-shaped device that glows in different colors. It can be configured to display data such as current energy prices or stock prices. Creator David Rose describes it as a “single-pixel browser [that] glows a color to reflect online information.” Light here isn't a secondary



signal—it's key for the primary purpose of the device: conveying information in a glanceable, ambient way, requiring less attention and providing less distraction.



**FIGURE 8-5**

The Ambient Orb uses different colored light to display information—for example, current energy prices (image: Ambient Devices)

The Ambient Umbrella (see Figure 8-6) uses light in a similar way. Its handle glows when it is going to rain to remind the owner to take it out.



**FIGURE 8-6**

The Ambient Umbrella has a light integrated in its handle that lights up when rain is forecasted (image: Ambient Devices)

LIGHTS	
<b>Are great...</b>	<b>... for glanceable and nonintrusive information output.</b> Whether it is about seeing something across a room, or integrating lights in a way where they don't distract, lights are great to give simple feedback without asking a lot of attention.
<b>Are less appropriate...</b>	<b>... for conveying complex information.</b> System status or error messaging can be complex with connected devices. With only few lights available, UX quickly becomes painful and complex (like interpreting blink patterns with the help of a manual).

### Visual input

Users can also give device input through light or other visual means. A device can use a camera to interpret a graphical pattern like a barcode or a QR code. Devices can also use light sensors to react to ambient lighting conditions or receive input from other devices through LED blink patterns.

Because these kinds of interactions and interfaces aren't often referred to as "visual input," we'll discuss them in a separate section ("Computer Vision, Barcodes, and Devices 'Seeing'").

### Screens and displays

Sometimes, you need more than lights. When a device needs to display complex information or users need to provide more sophisticated input, screens may be more appropriate. As a UX designer you'll be familiar with screen-based user interfaces of smartphones or personal computers. But there is a range of (simpler) display types that require different design approaches UX designers are less likely to be familiar with.

**Custom segment displays.** Custom segment displays are what many cheap electronic devices use, like the water filter system shown in Figure 8-7. The underlying technology is often LCD (liquid crystal display). This means transparent segments become visible when an electric charge is applied.



**FIGURE 8-7**

A custom segment LCD display; the overall display needs to be large enough to include all segments that might be shown (image: Martin Charlier)

When creating a product with such a display, every possible state the display needs to be able to show needs to be considered in the design phase. Even segments that might only appear in certain situations need to be given real estate. If a heating controller can show temperature in both Fahrenheit and Celsius, segments for both °C and °F need to be part of the segmentation layout. They cannot be in the same place.

Segment displays are low cost and use little power. But they are unable to display dynamic alphanumeric information. To do that, you can use a character set display.

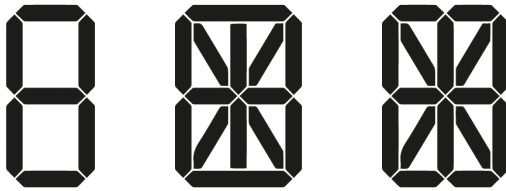
**Character set displays.** You most likely have seen a simple character set display in clocks or electronic calculators. These character set displays typically use a 7-segment layout for each character (see Figure 8-8).



**FIGURE 8-8**

A display using a 7-segment character layout (image: SparkFun Electronics)

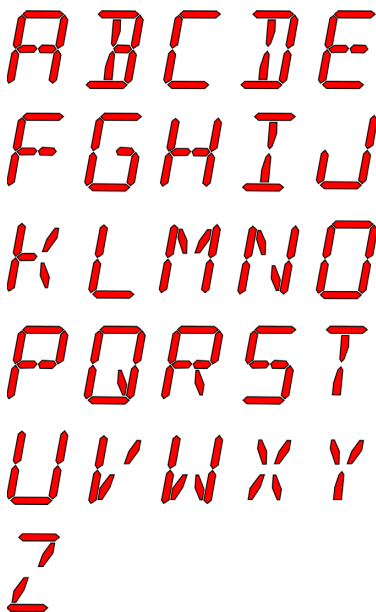
Character set displays work through systems of 7, 14, or sometimes 16 segments (see Figure 8-9) that are switched on in combination to create numbers. 14- and 16-segment displays can also display basic alphabetic characters.



**FIGURE 8-9**

The segment layout of 7-, 14-, and 16-segment displays (image: Martin Charlier)

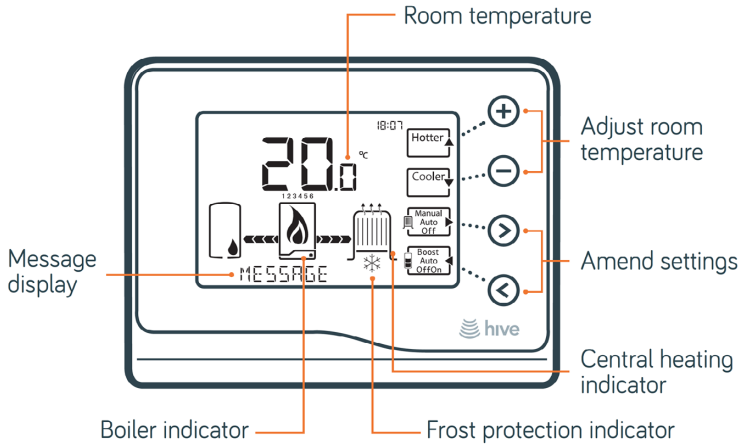
Displays like this are cheap, but have a limited character set. A 7-segment system is only usable for displaying numbers. Other types such as 14- or 16-segment systems can display letters, but with limitations. See Figure 8-10 for an example of the character set of a 14-segment display.



**FIGURE 8-10**

The character set of a 14-segment display (image: public domain)

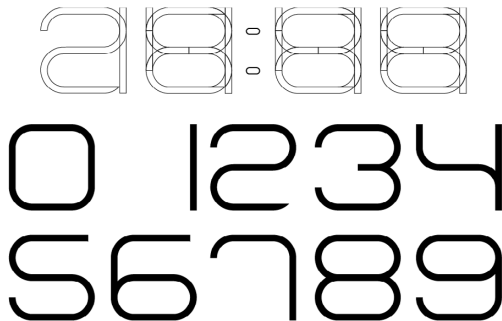
By combining character set and custom segment displays, you can display both dynamic alphanumeric data and illustrations or graphical elements. Figure 8-11 shows such a combination used in a heating controller. A custom segment display that shows boiler and radiator illustrations is combined with a character set display for dynamic messages.



**FIGURE 8-11**

Note the “Message display” area using a 14-segment display (image: British Gas)

Segmented character set displays don’t have to be ugly. The creators of the CST-1 watch designed a custom segmented font to display numeric characters (see Figure 8-12).



**FIGURE 8-12**

The segmentation used by the CST-1 watch; note that the first digit misses unnecessary segments, as it will only ever need to display “1” or “2” to display time (image: Central Standard Timing, LLC)

**Dynamic displays.** Displays with the fewest limitations have a matrix of pixels, each of which can be turned on or off individually. In principle, displays like this are the kind we're all used to from smartphones or computer monitors. A notable distinction is between monochrome and color displays.

Monochrome displays tend to be cheaper, but (as their name suggests) unable to display colors or shades. The Fitbit activity tracker uses such a monochrome OLED matrix display. It shows both alphanumeric information and simple graphics (see Figure 8-13).



**FIGURE 8-13**

The Fitbit activity tracker uses a monochrome matrix display (image: Fitbit)

UX designers will be familiar with color pixel matrix displays from smartphones, laptops, and tablet devices. Touchscreens are essentially pixel matrix displays combined with a transparent touch sensitive layer.

**Electronic ink displays.** A more unusual display technology with unique advantages and disadvantages is electronic ink.

Made popular through ebook readers like Amazon Kindle, the technology is used in more and more devices like connected watches or smartphones. Electronic ink displays resemble the appearance of traditional ink and paper, and are easy to read in direct sunlight. Unlike the previously described technologies, they only require electricity to change the state, not to hold it. This means the device uses less power.

This opens up new possibilities for designers, too. For example, an electronic ink display can show something when a device is off. To greet and guide users, a display could already show information when the user unpacks the product and hasn't even switched on.

There are also some disadvantages that constrain the design. Electronic ink displays have a very low refresh rate. They also suffer from an effect called “ghosting,” where shadows from a previous display state can still be seen. This rules out interactions like scrolling, the use of mouse pointers, or animations in the UI.

**Is a screen better than no screen?** You might think it’s good to put a screen on a device where possible, but this isn’t necessarily the case. Firstly, screens obviously increase the cost of a device. Having a screen means extra components and adds design and development effort.

Secondly, and perhaps more importantly, there are usability considerations, too. When designing apps or websites, a lot of work has already been done for designers. They can rely on platform features, design frameworks, and widely familiar interaction patterns. If you add a screen to a connected device, you essentially need to develop an interaction paradigm from scratch. How do menus work? How do you navigate them? How do users learn this potentially unfamiliar UI?

Moreover, once a screen is available, it becomes harder to say “no” to adding all sorts of information and features since real estate seems endless. So it becomes harder to keep devices simple and focused.

That’s why designers need to take a step back early on in the design process to ask: What if this product didn’t have a screen? Could it still work? You might just discover an opportunity for a beautifully simple and easy-to-use connected product.

SCREENS	
<b>Are great...</b>	<p><b>... for making physical objects dynamic.</b> For example, by combining screens with physical controls to get dynamic labels.</p> <p><b>... for keeping products flexible.</b> Using dynamic displays (and to some degree, character set displays) means the entire UI can be updated and changed through software. This enables updating and iterating the product even after it has shipped through firmware or software updates.</p>
<b>Are less appropriate...</b>	<p><b>... for keeping the user experience simple.</b> Once a product has a screen, the ability to display more information at no extra cost can lure design teams away from simplicity.</p>

## AUDIO AND VOICE INTERFACES

In this section, we'll discuss interfaces that use sound. This includes simple audio output using speakers, using synthesized voice output as well as interfaces that allow for user input through speech recognition.

### Audio output

Audio can be a useful output method. Users don't have to look at the device directly and don't even have to be near it. At the same time, anyone who has experienced the persistent beeping of a washing machine that has finished a cycle or a smoke alarm indicating its low battery knows how annoying audio output can be.

Audio is pervasive. Unlike lights, audio is almost never subtle. Sounds immediately trump visual indications. They are hard to ignore. Audio also has a bigger emotional character. This is obvious for speech: tone and accent can be perceived as sympathetic and friendly, or the opposite. It's also the case for other sounds: pitch and quality immediately carry emotions. A sound can convey urgency like an ear-piercing high-pitch beep. Or it can convey a sense of satisfaction like a fulfilling swoosh sound when an email is sent.

AUDIO OUTPUT	
<b>Is great...</b>	<b>... for urgent and time-critical alerts.</b> Because audio is pervasive, it is great to quickly grab a user's attention. <b>... to give a product emotional qualities.</b> The quality of sounds can be a great way to convey emotions. An audible alert can be designed to sound "efficient" or "playful," depending on what the designer wants to achieve.
<b>Is less appropriate...</b>	<b>... for environments where it becomes annoying.</b> The biggest challenge with audio is to not make it annoying. This requires a good understanding of the environment the product will be in and careful application of audio.

### Voice interfaces

We'll split voice interfaces into two subsections. First, using synthesized voice as output, and second, speech recognition interfaces that take voice as input.

**Voice as output.** Instead of sound effects, audio output can also be used for computer-generated speech. This is a powerful method for providing a wide range of information.



For example, the Nest Protect smoke alarm uses its speaker to play warning sounds. But it also gives more information using speech: a voice tells the user in which room smoke was detected.

In car navigation systems that use speech, the advantage is twofold: drivers can keep their eyes on the road, and information that would be visually complex can be simpler to convey in speech. An illustration of a road layout can become confusing when it doesn't exactly match reality, or when a junction involves a complex road layout. In such situations, verbal instructions don't necessarily become more complex. The nature of language means instructions need to be delivered in chunks (sentences) and sequenced (one after another, not all at the same time).

While this linear nature can be helpful in some situations (like the car navigation system), it has shortcomings in other contexts. For example, when voice output is used to present possible user commands (like in interactive voice response systems), its single information stream requires users to listen to and remember all possible options at a given time before making a decision.

Beyond usability, there are a few more things to consider if you're planning to use voice as a form of output, or a voice-based interface in general. Similar to the emotional aspects of sound we discussed earlier, the tone of voice or accent in a voice plays an important role too. This is a known consideration in the business of running call centers: for example, in the UK a study found that the Geordie accent is perceived as friendly and even as an accent that puts callers in a good mood.<sup>5</sup>

In addition, voice interfaces with synthesized speech might be constrained to certain geographic regions by virtue of only supporting selected languages. For example, at the time of writing, more than 3 years after Apple Siri was first introduced into the iPhone, it still only supports 9 different languages and 15 regions.

**Voice as input.** Speech recognition is available on most modern smartphones. When it works, it can be a quick way to give complex commands. Setting an alarm using a touchscreen, for example, requires a series of steps that include fiddling around with UI components to

---

<sup>5</sup> "Geordie Accent Rated UK's Most Friendly, Most Likely to Put You in a Good Mood," Bloomberg, April 20, 2010, accessed March 25, 2015, <http://bloom.bg/115LLh8>.

select hours and minutes or even doing math to translate the user’s way of thinking (“wake me up in 3 hours from now”) into the constraints of the UI (set time when alarm should go off). A voice command for the same input can be quicker and easier (“wake me up in 3 hours”).

Voice input can remove the need for many other controls. The Amazon Dash (see Figure 8-14), a device that lets users add items to a shopping list, only has two buttons: one to activate voice input, and one for a bar-code scanner. Voice input replaces an entire keyboard in this case.



**FIGURE 8-14**  
A person speaking into Amazon Dash (image: Amazon)

But there are also a few challenges with the technology.

Speech recognition is still unreliable. Anyone that has used it has encountered situations where commands were misinterpreted. Feedback mechanisms are required to confirm correct recognition. This can be done using a screen, like in interfaces such as Apple Siri or Google Now. When the point of using voice as input is to avoid looking at a screen, of course, this feedback mechanism isn’t available. Confirmation can be done through speech output, but this quickly becomes a frustrating loop going back and forth.

Another challenge is to know what commands are possible in the first place. Users must recall (instead of recognize) what “functions” are available, or how they need to phrase commands. Devices with screens often display a list of possible commands to help users with this problem.

Devices need to know when to listen for a command. One approach is to press a button, but this means the overall interaction isn't hands free. Another approach is activation phrases like "OK, Google" that indicate that a command follows. The downside of this approach is that users agonize over the idea that a device is constantly listening to everything that is being said. It also makes false positive detection a problem—when you say something that sounds similar to "OK, Google."

A prank exploiting this is to play Xbox online multiplayer games using the gamer name "Xbox sign out." Other players will eventually say the name out loud, perhaps to vent their confusion about it in the game's audio channel. If those players have the voice control features enabled, they will soon disappear from the game server.<sup>6</sup>

Lastly, most of the current voice recognition features rely on server-side processing to interpret what a user has said. This means the feature relies on the device having a reliable and fast data connection, which fundamentally limits the number of situations where it can be used.

VOICE INTERFACES	
<b>Are great...</b>	<p><b>... when other channels are occupied.</b> Think car navigation. Drivers can keep eyes on the road and still get information.</p> <p><b>... to give complex information on an otherwise simple product.</b> Think the Nest smoke alarm that uses speech to say where in the house smoke was detected.</p> <p><b>... for products where complex data input is required.</b> Tasks that take a number of steps on a touchscreen UI, like setting an alarm clock, are much simpler and faster using speech recognition.</p> <p><b>... when a minimal physical interface is desired.</b> Although a device might be capable of complex input, the impact on industrial design and the physical appearance can be minimal.</p> <p><b>... for contexts where hands-free interaction is desired.</b> For example, in the kitchen, to keep working while interacting with a device. Also to keep the device sterile because it isn't touched.</p>

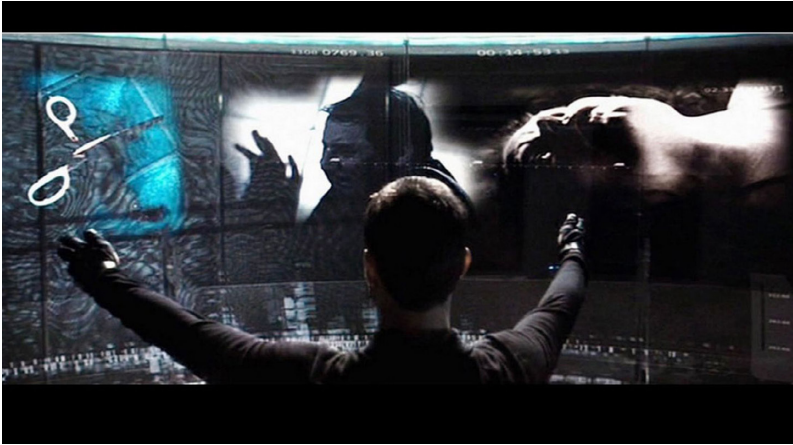
<sup>6</sup> Xbox One Sign Out Trolling: <https://www.youtube.com/watch?v=mWZLa4AnN5k>.

<p><b>Are less appropriate...</b></p>	<p><b>... for products or contexts where weak connectivity is possible or that are used in high noise environments.</b> Speech recognition is often done server-side. A fluid experience requires a good data connection. Noise or other people talking make recognition less reliable.</p> <p><b>... when the nature of commands and terms used is hard to pronounce or recognize.</b> A speech-controlled music player is likely to struggle with unusual artist and song names, not necessarily due to recognition, even just due to the fact that correct pronunciation might not be clear to users.</p> <p><b>... for products aimed at many different global markets.</b> Not all languages might be reliably supported. A fast global rollout might be impossible. At the time of writing, even Apple's Siri is not globally supported.</p>
---------------------------------------	--

**GESTURAL INTERACTION**

Gestural interactions exist on touchscreens (like “swiping” or “pinching”). Using computer vision, devices can also recognize mid-air gestures as user input.

Perhaps one of the most famous examples is from the film *Minority Report*. Chief John Anderton (played by Tom Cruise) controls a computer system using both his hands in mid-air, almost like conducting an orchestra (see Figure 8-15).



**FIGURE 8-15**

The film *Minority Report* made gestural control popular although it is an interface method with many flaws (image: Twentieth Century Fox Film Corporation)

The film has had major influence on the development of gestural interfaces. But research has since shown that gestural interfaces might not be appropriate for real-world applications. Holding out your arms in front of your body, gesturing, or doing precise motions for extended periods of time causes aching muscles and stiffness. This condition has been dubbed “gorilla arm.” Rumor has it that Tom Cruise had to take breaks from even just acting the scenes for the movie.

Yet, industries like gaming have adopted gestural interfaces with great success. The Microsoft Kinect lets gamers control Xbox games or navigate menus without any further device or controller. Users wave hands or move their body to control what they want to do.

Simpler forms of gesture interaction have also been integrated into more mundane products. The original Nest Protect smoke alarm had a feature called “wave to hush.” The feature was meant to solve the problem of alarms going off because of burnt toast or smoky frying. Users could stand underneath the device and wave their hands at it to silence it for a short period of time (see Nest’s illustration in Figure 8-16).



**FIGURE 8-16**

The now disabled “wave to hush” feature let users silence the smoke alarm by waving their hand at it (image: Nest Labs)

In gestural interfaces, there is always a risk of false positive input detection. In the case of the Nest “wave to hush” feature, the possibility of users unintentionally activating it (perhaps waving their hands during a real fire) could delay an alarm. In April 2014, Nest had halted sales and disabled the feature on already installed units to investigate potential problems with the feature. So, gesture interaction can be an elegant solution to interacting with a device that is hard to reach (like a smoke alarm). Yet, designers need to think carefully about the context and risk of false positives.

### Intangible interactions

It’s worth briefly discussing some fundamental differences in interfaces where user input is more intangible, like in gesture or voice recognition interfaces, to interfaces where input is very explicit. For intangible interactions, the space that the interface needs to sense to recognize user action is necessarily quite broad: for example, all movement in a device camera’s field of vision or all sounds audible to the device’s microphone. Input is inferred from actions happening in that space, as opposed to being explicitly received. A device controlled through buttons receives explicit input when a button is pressed and so it only has to “sense” a contained space that is the state of that button. An intangible interface like a gesture interface “senses” all movement in front of its sensors and infers input from identifying particular gestures or patterns. This means there is always more room for error and false positive input.

GESTURAL INPUT	
<b>Is great...</b>	<p>... <b>for video games.</b> Gestural control makes games more immersive and physically challenging.</p> <p>... <b>for short interactions.</b> The “gorilla arm” becomes a problem with lengthy use. If gestural interaction is kept short this is less of an issue.</p> <p>... <b>when it is obvious what gestures are possible.</b> Users need to know or recall what gestures are possible, or have a guide.</p>

<p><b>Is less appropriate...</b></p>	<p><b>... when precision and lengthy interaction is required.</b> Giving precise input is hard with gestural interfaces. Lengthy interaction leads to fatigue and muscle pains.</p> <p><b>... when false positive recognition can have serious consequences.</b> Recognition is still unreliable and false positives can occur. Gestural input is not appropriate if this can have serious consequences.</p> <p><b>... when there is no time to learn.</b> If the application doesn't allow for a learning phase to let users understand how to use the interface, other interface types might be more appropriate.</p>
--------------------------------------	---

## TANGIBLE AND TACTILE INTERACTION

In this section, we'll discuss two types of interfaces: ones that work through manipulating tokens or placing a product in a certain way to give input (we'll call these tangible user interfaces), and interfaces that involve haptic or tactile feedback like vibration or force feedback.

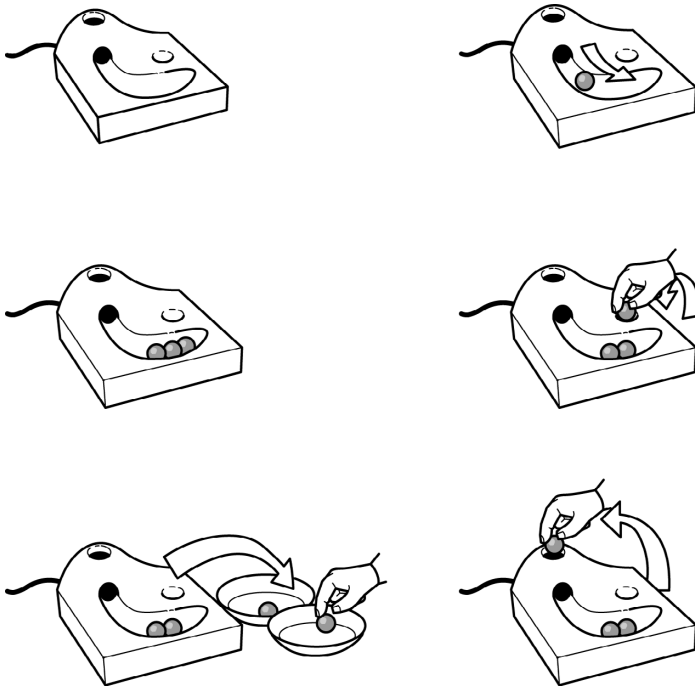
### Tangible user interfaces

Earlier in this chapter, we discussed interfaces using physical controls like buttons, sliders, or dials. Such physical controls are certainly a type of tangible interface and so are keyboards or computer mice. In this section, however, we look at tangible interfaces where input is given through manipulating physical objects or tokens representing intangible assets.

In 1992, designer Durrell Bishop created one of the earliest explorations of this idea. Bishop envisioned an answering machine that made messages tangible through marbles that represent messages (see Figure 8-17).

Colored marbles represent recorded messages. The machine releases them into a groove for the user to pick up. To play a message, the user places a marble into a depression on the surface of the machine. Playback stops as soon as the user removes the marble from that spot. Users can either keep messages by holding on to the marbles, or delete them by putting the marbles back into the answering machine through a hole.

A more recent exploration of tangible interaction is designer Jordi Parra's Spotify Box, which was created as a student project (see Figure 8-18).



**FIGURE 8-17**

Illustration of the marble answering machine. Top right: New message; Middle left: Three messages; Middle right: Play message; Bottom left: Leave message for flat mate; Bottom right: Delete message (image: Durrell Bishop/Durrellbishop.com).

The box plays music streamed from Spotify. By placing a physical magnetic token on the box, users can select a playlist. No menu or button combination is required. This gives an analog quality to the interactions with digital music. Users can label tokens or draw on them. They can also exchange them with other owners of the Spotify Box.

The marble answering machine and the Spotify Box are interesting concepts because they explore giving things that are intangible (e.g., voice messages or music) a physical representation that users manipulate or control physically. This changes the nature of interaction with a digital system, turning it into something more direct and immediate.

Tangible interfaces offer a way to simplify the control of a device. The direct and physical nature of manipulation can be easier to learn and understand than abstracted menu structures or conceptual mappings of controls.





**FIGURE 8-18**

Jordi Parra's Spotify Box with a playlist token attached to the “play” spot (image: Jordi Parra)

Sifteo cubes were a great example of this, but they are unfortunately no longer available (see Figure 8-19). They are matchbox-sized cubes with a screen that sense when they are next to each other. Through tilting, placing, or rearranging them, kids can play educational and entertainment games across the cubes.



**FIGURE 8-19**

Sifteo Cubes sense when they are next to each other to enable games played by manipulating and placing the cubes (image: Sifteo)

A simpler example is the Muji Multi Clock alarm clock. It's a cubic device with labels for its four functions placed along each side of its display. Users select a function by orienting the clock with the corresponding label on the top. Figure 8-20 shows the calendar function activated.



**FIGURE 8-20**

The Muji Multi Clock (no longer sold) with the calendar function activated (image: Muji)

A promising application of tangible user interfaces is in musical instruments. The Reactable (see Figure 8-21) is such a musical instrument with a tangible user interface. Musicians manipulate and create sounds using physical objects placed and moved on the table's surface. The tangible nature means the musician can create and control many different sounds and keep the overview during a live performance.

Experimenting with tangible user interfaces is thought provoking and inspiring. But interfaces involving many small physical tokens to manipulate or represent data have an obvious impracticality: if you lose parts, the interface breaks.

Imagine not being able to listen to messages because your cat or a child has made off with the marbles. So you need to think carefully about the application of tangible interfaces.



**FIGURE 8-21**

A Reactable in use making music (image: Reactable/Massimo Boldrin)

TANGIBLE USER INTERFACES	
<b>Are great...</b>	<p><b>... for digitally enabled experiences that don't feel like interacting with a computer.</b> For example, museum exhibits that let visitors explore and experiment, or musical instruments like the Reactable that lets users interact with sounds and music.</p> <p><b>... for educational products.</b> Learning through manipulatives is an important aspect of early childhood education and approaches such as Montessori education. Tangible user interfaces can keep learning experiences tactile while enhancing them with dynamic digital content.</p>
<b>Are less appropriate...</b>	<p><b>... when keeping all parts together is critical.</b> If lost parts mean lost data or lost access, the drawbacks of tangible user interfaces might outweigh their benefits.</p> <p><b>... when there is no time to learn.</b> If the application doesn't allow for a learning phase to let users understand how to use the interface, other interface types might be more appropriate.</p>

### **Tactile output: Vibration, force feedback, and shape shifting**

We've all experienced vibration alerts of our mobile phones. It's a useful feature when it's noisy or when sounds would cause disturbance. But it requires body contact, which few devices have reliably.

The potential of tactile output is intriguing: researchers at the University of Amsterdam have explored the use of vibration for cyclists. Their “vibrobel,” a GPS-based navigation using vibration, is worn around the waist and gives the cyclist tactile feedback when they need to take a turn.<sup>7</sup> Apple’s watch has a similar feature. It can “tap” users on the wrist during navigation, using different taps for “go left” or “go right.”

A much more sophisticated form of tactile output are force feedback mechanisms. These use motors to push back on the user in specific directions. Gaming joysticks use this to great effect to let players feel the g-forces involved in flying a plane, for example.

A more serious device using force feedback is the 3D Systems Touch (see Figure 8-22). This is a device that makes virtual objects tangible.



**FIGURE 8-22**

The 3D Systems Touch uses force feedback to make virtual objects tangible (image: 3D Systems)

The central feature is a pen-like device mounted on multiple motorized axes. Users can give input in three dimensions using the pen, but the device can also “push back” in all three dimensions, too. The applications are specific, such as 3D CAD modeling or surgery robots. An advantage is that users can physically feel virtual boundaries.

---

7 See this link for more: <http://bit.ly/115Oszh>.

As with tangible interfaces, it's intriguing and inspiring to explore how tactile interfaces can change everyday interactions. The Proverbial Wallets project from MIT explores this with financial data. The researchers created three prototype wallets, each using a different kind of tactile output for different kinds of data.

The first wallet is Bumblebee. It uses vibration to notify the owner of every transaction that is happening with their bank account. The second is Mother Bear (see Figure 8-23). It uses its force feedback hinge mechanism to make it harder (or easier) to open the wallet depending on the current budget. The last is Peacock. It's a wallet that swells and shrinks, reflecting the balance in the owner's account.

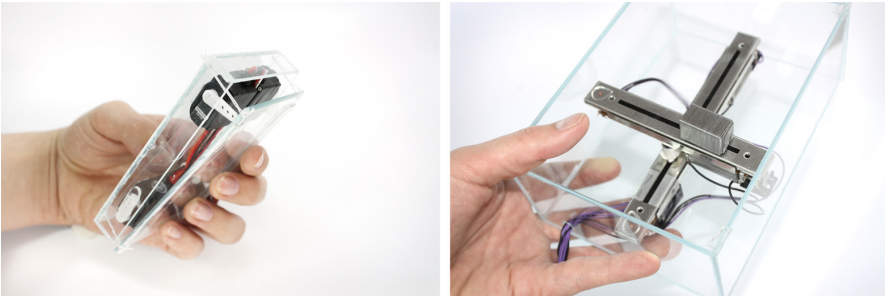


**FIGURE 8-23**

One of the three prototype wallets from the Proverbial Wallets project (image: John Kestner, Supermechanical)

Researcher Fabian Hemmert has explored the possibilities of tactile output in mobile phones. His prototypes include devices that shift their center of gravity and devices that shift their shape (see Figure 8-24).

As with the Proverbial Wallets, it is intriguing to think about the ambient representation of data. Instead of displaying it in concrete terms, users feel it while interacting with the device. Hemmert gives the example of representing progress through a playlist and the current song at the same time using the center of gravity, or giving nonvisual directions in navigation applications.



**FIGURE 8-24**

Fabian Hemmert's prototypes for shape- and center of gravity-shifting devices (images: Fabian Hemmert)

TACTILE OUTPUT	
<b>Is great...</b>	<b>... for creating interfaces that demand less attention.</b> Tactile output can reduce cognitive load. Instead of interpreting visual output, it can be faster (but less precise) to <i>feel</i> the information.
<b>Is less appropriate...</b>	<b>... (currently) for affordable, reliable systems.</b> Mechanical systems involve friction, wear and tear, and they have a limited lifespan.

## CONTEXT-SENSITIVE INTERACTION

Context, or “the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood” (Oxford Dictionaries), plays an important role in the design of almost anything. Throughout this chapter, and the entire book, we discuss when some interfaces are more or less appropriate, how user scenarios and situations inform design decisions you make, and many other things that all, in some ways or others, are about understanding context.

In human–computer interaction, understanding and using context, or “context-aware computing,” is an area of research in its own right with many papers, experimental projects, and healthy discussions. The definition of “context” in context-aware computing varies across articles and papers written about this, but the general idea is that computers sense and act on inputs such as time of day, the weather, the current location, or the identity of the user.

In examining what makes up context, you’ll find that there are many different kinds of contexts that play a role in the design of a product and the user’s interaction with it. For example, there is social context (i.e.,

who a user is or who they are with), environmental context (i.e., where a user is and whether this place is noisy, rainy, outdoors, or indoors), and user activity context (i.e., whether the user is preoccupied with another activity like driving or cooking).

These examples are loosely based on some of the seven “flavors” of context designer Cennydd Bowles discusses in a post on his blog.<sup>8</sup>

In systems where apps or websites form part of the controls, designers can rely on a range of inputs for contextual information (e.g., device location, user identity, social network data, calendar entries, etc.). The British Gas Hive app, for example, uses device location to determine when the user has left home and sends a push notification if the heating is still set to higher than a predefined temperature.

When it comes to connected physical products, context sensing is still fairly limited. Some devices use proximity sensors or GPS to sense presence or location, but many facets of context aren’t currently used to tailor and enhance the user experience.

The opportunity of context-sensitive interaction is to enable products that behave in ways that fit better into the user’s routine and behaviors—for example, by transforming active user input into more passive input. Instead of gesturing to make a command, simply being (or not being) in a certain room can provide input to a system, too. Another example is being close by or further away from a product. This can be useful information to tailor an interface.

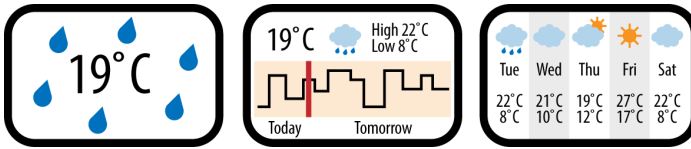
David Rose, founder of Ambient Devices, demonstrated a product that works in such a way at the LIFT’09 conference. Rose showed a connected weather forecasting device with a screen and a proximity sensor.<sup>9</sup>

Using the proximity sensor, the device changes the amount of information it displays (see Figure 8-25).

---

8 “Designing with context,” Cennydd Bowles’ Blog, accessed March 25, 2015, <http://www.cennydd.com/blog/designing-with-context>.

9 See the talk here: <http://bit.ly/1I5Pb3r>.



**FIGURE 8-25**

Example illustration of a weather forecasting device that displays more or less information depending on the proximity of a person in front of it (illustration by Alfred Lui)

When a user is standing across the room, the screen displays only essential information: the current outside temperature fills up the screen in large digits and the background gives clues about the weather (e.g., rain drops around the temperature). When the user is closer to the device, the screen shows more information using smaller fonts. It now includes a forecast for the next 24 hours. Finally, when users tap the device, it shows even more information.

The interface is sensitive to proximity, which is far from sensing all there is to the user's context, but it allows for some basic interpretation: If the user is across the room, they might benefit from the product providing information in a condensed way that is readable from a distance. If they are standing right in front of the product, font sizes can reduce and the user might benefit from more detailed information. Note that this interpretation of physical context into user intention is just that: the designer's interpretation of a sensor's reading as a proxy for something else. In other words, the product doesn't really sense context; rather, it senses proximity—and it's important not to forget that.

The Aether Cone connected speaker (Figure 8-26) also has some context-sensitivity features. It's a mobile device that learns about the user's musical preference in different rooms of the house. It uses this to automatically adjust what it plays based on what room the user places it in. It is important to remember that the device builds up a picture of two things:

- Its own location
- The user's actions to change what is playing



The insight, or hypothesis, that users listen to music differently in different rooms, and that these two data points can be combined to provide a better experience based on this idea is the designer's work and may or may not be true for the user. Other contextual information that is harder to sense is likely to influence the user's musical preference, too. For example, whether the user is alone or with other people might change the musical preference, too.



**FIGURE 8-26**

The Aether Cone connected speaker remembers what music the owner prefers to listen to in different rooms (image: Aether)

An opportunity for context-sensitive interfaces is to demand less of the user's attention and require less interaction. Ideally, a device could take actions for the user, or tailor the user experience, because it understands the context and what the user would desire. This is a slippery slope and finding the right balance between taking action on behalf of the user and keeping the user in control needs to be found.

Designing interfaces like this requires a very good understanding of your users. Designers can only ever assume the user's context through a proxy that a device can sense, like location or proximity. It's the designer that needs to establish the correlation between such data points and user intentions or desire. Careful user experience design then needs to ensure that any context-sensitive features don't worsen the user's experience if the assumptions made aren't true.

CONTEXT-SENSITIVE INTERFACES	
<b>Are great...</b>	<p><b>... to manage levels of complexity with little interaction.</b> Contextual interaction can be a great to move between levels of detail without requiring methodical user input.</p> <p><b>... when there is an intuitive or obvious relationship between contexts and device features.</b> For example, approaching a device and revealing information are easy to connect conceptually.</p>
<b>Are less appropriate...</b>	<p><b>... when it might be perceived as patronizing.</b> Limiting options or taking action on behalf of the user might be undesired and perceived as negative and overbearing.</p>

### COMPUTER VISION, BARCODES, AND DEVICES “SEEING”

In this section, we’ll talk about interfaces that involve devices seeing or responding to artifacts and activity around them. By this we mean interactions such as face, object, or pattern recognition, as well as other ways a device can receive input—for example, by scanning barcodes or receiving light input.

To clarify, computer vision isn’t an interface type but an enabler. The gestural interactions we discussed previously rely on computer vision. Simpler forms such as face recognition can enable useful features such as smartphones that can unlock using the owner’s face. Face detection (instead of recognition) can enable smart cameras that automatically follow a person’s face.

The power of these kinds of interactions is to simplify or remove user input. Compared to inputting a PIN code, it’s faster and easier to just look at a device to unlock it. It may be less secure, but that is a different story.

This power of simplification is also handy when users need to enter information from the physical world (like a book title) to start a digital interaction. The Amazon Fire Phone introduced a feature called FireFly that does just that. In Figure 8-27 you can see the device recognizing a physical book in front of it. It then displays relevant information and actions.

In connected products, designers can use these kinds of interfaces to tailor the experience—for example, by recognizing who is in front of a device. A challenge here is not to create a patronizing experience.



**FIGURE 8-27**

Amazon's Fire Phone can recognize objects and present relevant shortcut functions (image: Amazon)

The smart drinks vending machines installed in some train stations in Tokyo are a real-world example that caused some debate due to their Orwellian characteristics. Instead of a glass through which available drinks can be seen, this machine displays its selection using a large screen on its front. The machine also uses a camera to determine the age and gender of whoever is standing in front of it. Based on this information, the machine displays a targeted selection of drinks. For example, it is said to suggest a slightly sweeter beverage to a woman in her 20s.<sup>10</sup> The device also captures the data and the actual choice the customer made for marketing analytics. To some users, even the idea of a vending machine prescribing choices, trivializing social norms, and collecting user data is shocking. The sensitivity to features like this is different in different cultures. Allegedly, the Japanese customers don't share such views and concerns.<sup>11</sup>

10 Julian Ryall, "Japanese vending machine tells you what you should drink." *The Telegraph*, November 16, 2010, accessed March 25, 2015, <http://bit.ly/1I5PY4q>.

11 "Japanese embrace "human-feeling" machine." *Globalite Magazine*, October 10, 2010, accessed March 25, 2015, <http://bit.ly/1I5POdk>.

Marketing applications aside, there are also practical uses of such interfaces. A well-known example is QR codes. It's a useful way for users to input complex information with a minimal interface. Instead of manually entering WiFi credentials, for example, users could present them using a QR code for a device to decode.

Scanning a QR code requires a camera and the ability to interpret the image presented. This means the device to scan needs to be relatively complex. But the principle of encoding and transmitting information optically can take other, less computationally intensive ways, too.

The Electric Imp, a platform to connect a device to the Internet uses a form of optical communication to receive WiFi credentials. The Imp uses a simple light sensor (instead of a camera) to receive this input. Users enter the WiFi credentials on a smartphone that turns them into a sequence of light flashes performed by the screen. The light sensor recognizes the sequence and the device decodes the information contained. Electric Imp calls this process “BlinkUp.”

The connected kids' toys manufactured by Toymail use this platform to connect. See Figure 8-28 for the key steps in the process.



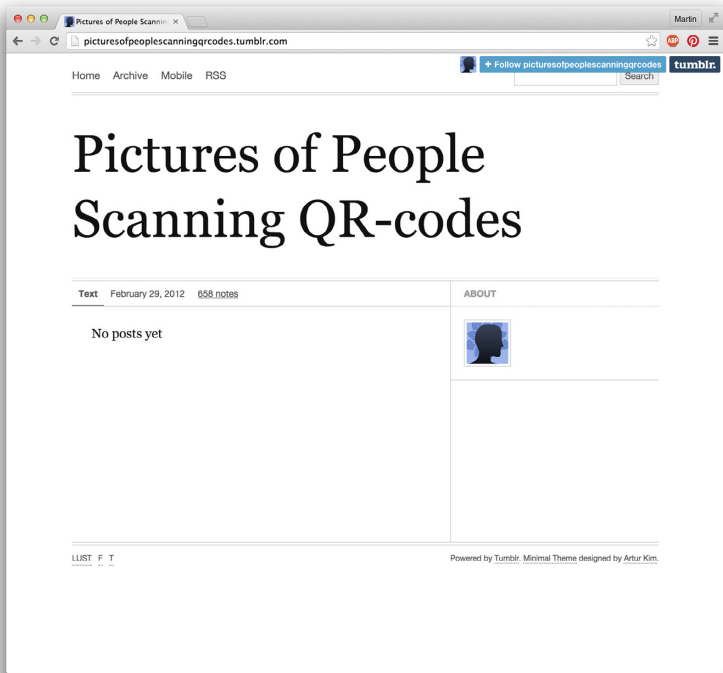
**FIGURE 8-28**

WiFi credentials are entered on a smartphone, the device is placed on the smartphone screen with its feet, the screen flashes on and off for a short period, an LED in one of the feet lights up green when the process succeeded (image: Toymail Co.)

### **Are QR codes good or bad?**

QR codes have a bad reputation among designers. The apps required to scan them haven't reached widespread adoption. The outcome of scanning one often doesn't match the effort required to scan a code in the first place. An Internet joke mocking this reality is a Tumblr blog that

has been set up to collect pictures of people scanning QR codes. The irony is, since its creation in early 2012, no pictures have been posted to it (see Figure 8-29).



**FIGURE 8-29**

Pictures of People Scanning QR codes—Tumblr page (image: Martin Charlier)

But when the scanning logic isn't for the user to worry about, QR codes are a perfect means to input complex data quickly. Great examples are airplane boarding passes or event tickets stored on mobile phones. For connected products, they can be a great method, too. A connected lock might use its door camera to validate a temporary key presented using a QR code.

COMPUTER VISION AND BARCODES	
<b>Are great...</b>	<b>... for replacing cumbersome input.</b> Computer vision and barcodes can replace manual input like typing in WiFi credentials, URLs, or book titles.
<b>Are less appropriate...</b>	<b>... when the interaction is more complex than its alternatives.</b> If additional software (like a QR code reader) is required, users may view the alternative interaction as simpler.

### MULTIMODAL INTERACTION AND INTERFACE COMBINATIONS

When we talk to another person, we communicate using multiple channels at the same time. Speech, gestures, facial expression, and body language work together. In a similar way, interaction with a connected device can combine different types of interfaces into one. This is called multimodal interaction.

A great example of this is Jared Ficklin’s Room-E project<sup>12</sup> (see Figure 8-30). By combining gesture and voice interaction, Ficklin demonstrates how users can provide input similar to the way they would talk to another person. In his demo, Ficklin points at a specific light in the room (gesture input) and says “Computer, turn off this light” (voice input). The computer combines both inputs into a single command. It is able to understand that Ficklin specifies which light he means to switch off using the gesture.<sup>13</sup>

Although Room-E is a prototype, it shows the potential of multimodal interaction. By combining interface types, designers can create interactions that feel more like human-to-human interaction.

### IoT-Specific Challenges and Opportunities

Having discussed different interface types from a relatively general point of view, we will now take a look at some of the specific challenges to do with device interfaces you might face in developing a connected product.

<sup>12</sup> Jared Ficklin is a frog fellow and the Chief Creative Technologist at argodesign. Find him on Twitter (@jaredrawk) or at [argodesign.com](http://argodesign.com).

<sup>13</sup> Watch the demo here: <https://www.youtube.com/watch?v=d-LvtHs-6Oo>.



**FIGURE 8-30**

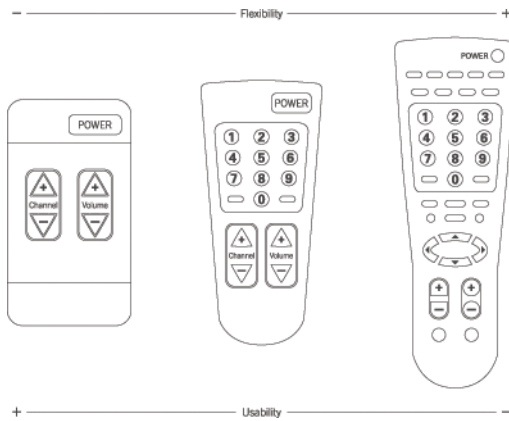
Jared Ficklin demonstrating Room-E; a gesture (pointing at a lamp) is combined with voice input (“Computer, turn off this light”) to control the room (image: Frog Design / Taylor Hamilton)

## **DECIDING ON THE LEVEL OF INTERACTIVITY OF A CONNECTED DEVICE**

Adding interactivity to a device is a cost question. Buttons, switches, and the components required add to the bill of materials and increase development costs. It’s also a usability question. Designers need to strike a balance between easy-to-use devices and ones with many features and functions. This is sometimes referred to as the “flexibility usability tradeoff” (see Figure 8-31).

However, simple devices with little features have other disadvantages. They often rely on other devices for full-feature interaction. This makes them dependent on things outside of the designer’s control, such as a working Internet connection or a compatible browser. Imagine you couldn’t change the temperature in your house because of problems with your Internet service provider.

An example of this approach was the first Tado thermostat. Users controlled it using a website and smartphone app. But the product offered a minimal degree of control on the device, too. A single physical button let users switch the heating on or off as a fallback option.



**FIGURE 8-31**

The flexibility usability tradeoff as illustrated in the book *Universal Principles of Design* (image: *Universal Principles of Design*, Rockport Publishers, 2010)

Tado has since introduced a new generation that features capacitive touch buttons and a display to adjust temperature up and down right on the device (see Figure 8-32).



**FIGURE 8-32**

The first Tado thermostat had only one button and relied on a smartphone app for full control; the latest generation device includes a little more control on the device via two additional capacitive touch buttons (image: tado° GmbH)

Only having few controls on a device can be advantageous if you know your product will change and develop. In this case, supporting a physical interface can become a burden when system features are changed or abandoned.



There are ways to balance the benefits and drawbacks of physical controls. An example is Cloudwash, BERG's connected washing machine prototype (see Figure 8-33). The physical device has some controls that only cover selected functions. Full control is delivered through a smartphone app. But users can perform the everyday tasks without needing the smartphone app. Also, the device uses a combination of physical controls with displays. This keeps the physical interface flexible within reason.



**FIGURE 8-33**

BERG's Cloudwash prototype uses displays to dynamically label physical controls (image: BERG)

From BERG's website:

The legends for these washes are in e-ink so they can be updated but still feel like part of the legends one would expect to find on a washing machine. They can be far more descriptive than the descriptions on a traditional dial, since there's no need for generalisations like "cottons." We like human-readable descriptions like "Joe's sports kit" or "really really quick" or "use this for baby stuff."

## MOBILE AND WEB UIS

A lot of connected products use mobile apps or web interfaces as their primary means of control or split the interaction between them and the device itself. There are good reasons to do this:

- App and web interfaces are much easier to change and develop in response to changing or developing requirements. If you add or remove a feature, updating or rethinking the UI of an app or a web interface accordingly is less of a problem than when you have a fixed interface on an embedded device.
- App and web development have a large, active developer community with many useful resources easily available. There is a lot of documentation for best practice solutions and even libraries developers can use to solve particular problems as opposed to developing from scratch.
- Smartphones and personal computers provide ample processing power to allow for features an embedded device might not be capable of (like speech recognition or using a camera). High-resolution color screens and rich input capabilities make it easier to create interfaces that move between simple and complex when necessary. (For example, allowing text input when necessary, but not bloating the interface with a keyboard for most of the time.)
- Users carry smartphones with them, so user location can become a trigger or data point used by the product. Because a smartphone is a personal device, interfaces can be tailored to a particular user.
- App stores handle the distribution of software updates so developers don't have to.
- App platforms offer notification channels that are irrespective of the user's location or that of the connected product. This means a connected product can notify a user even if they aren't near it.

But there are also reasons why you might avoid relying on a mobile phone or web UIs:

- Smartphones aren't reliably available all the time: users might switch them off or turn them to silent mode, or the devices might run out of battery or lose a signal. This all makes smartphones less appropriate for critical notifications that need to be received immediately.
- Web UIs that require a personal computer, but also using a smartphone can be slower than on-device controls in the right context. For example, some users of connected locks complain that the

process of getting out their smartphone and launching the right app takes them longer than getting out their keys and unlocking a door.

- When functions or alarms need to be available to anyone in certain locations, whether they have a smartphone or not, on-device interfaces are necessary.
- Less tech-savvy users can have difficulty building accurate mental models of how a system works. For example, during the research for BBC’s Radiodan project, participants connecting to the radio’s setup interface using their web browser were unsure if the radio would still work when they closed the browser window.

Even if not appropriate for all connected products, using web UIs and mobile phones especially is a great way for designers to off-load complex and rarely used features, and to keep the physical device simple. Chapter 9 discusses in depth the design decisions to be made in distributing interactions between different interfaces on different devices.

Specific design advice for how to design for mobile or web UIs is not within the scope of this book, and there are some great resources available on these topics:

- *The Mobile Frontier* by Rachel Hinman (Rosenfeld Media)
- *Don’t Make Me Think* by Steve Krug (New Riders)
- *Designing for Interaction* by Dan Saffer (New Riders)
- *Designing Interfaces* by Jenifer Tidwell (O’Reilly)

## **GLANCEABLE AND AMBIENT INTERFACES**

We’ve already touched on how designers can create interactions that require less attention. Considering the ever-increasing amount of information around us, this will become ever more important. We’ll soon all crave a less distracting environment.

This kind of interaction is already around us. Consider how a wall clock “disappears” into our surrounding. We completely ignore it for the most part. The moment we’re interested in the time, we can get it at a glance. Even from across the room. We might only be able to see roughly the angle of the two main hands—but it’s enough to give us an idea of what time it is. If we want more detail, we can walk closer to see exactly what time it is.

The ambient devices mentioned earlier are attempts to achieve similar characteristics. That can mean devices that only grab the user's attention when required, like the Ambient Umbrella, or devices that convey information in a way similar to how you sense what time of day it is by seeing, in the corner of your eye, how bright it is outside through a nearby window. David Rose, founder of Ambient Devices, has created a list of these characteristics that designers can use to inform and inspire the design of their products:<sup>14</sup>

*Pre-attentive*

Glanceable; no cognitive load required

*Calm*

Nonintrusive; seamless with environment (e.g., Evergreen, Friendly)

*Universal*

No language, characters, or numbers

*Open*

Able to represent multiple types of data; coded, private

We highly recommend exploring how you could bring some of these characteristics into the products you're working on, even if it's just a thought experiment.

## **WORKING WITH LIMITED INPUT AND OUTPUT CAPABILITIES**

A particular challenge in connected devices is to work around limited input and output capabilities. Many connected devices only require complex input rarely and a simple interface is sufficient most of the time. So how can designers handle these rare, but complex interactions without having to bloat the interface with capabilities that aren't needed most of the time?

The Withings Smart Body Analyzer provides a great example. The smart body scale supports multiple users, each with their own accounts. It automatically identifies which user is standing on it by comparing the reading with its historic data. If the new reading is within a realistic extrapolation of only one account, it can only be that person.

---

<sup>14</sup> See slide 51 here: <http://www.slideshare.net/davidloringrose/designing-ambient>.

This doesn't always work. When a reading is within range of more than one account, the device cannot reliably decide which user it is. In this case, the device needs user input. But instead of having specific controls for this situation, the designers found a solution that keeps the interface simple. They work with the capabilities available. In this case, the display shows a selection between the likely accounts. Users then lift either their left or the right foot to make a selection (see Figure 8-34).

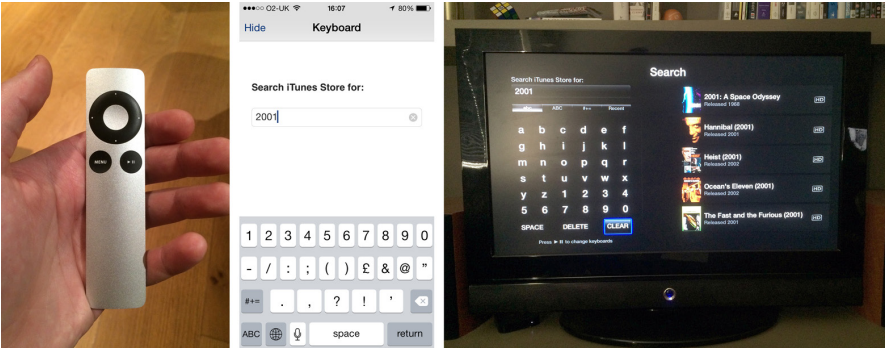


**FIGURE 8-34**

The Smart Body Analyzer displays a choice between two users when it is unsure who the current reading belongs to; by standing only on the right or left side of the scale, the user can make a selection (image: Withings)

Another strategy is to support devices with more capabilities. Navigating and controlling Apple TV with the minimalistic remote is sufficient for most of the time. When it comes to entering search terms or login credentials, using the remote becomes tedious. By supporting an iPhone as an alternative input device, users can switch to the more capable device when required (see Figure 8-35).

The Amazon Fire TV remote (see Figure 8-36) uses speech recognition to overcome this same hurdle. Its remote has a dedicated button for voice search and users can speak a search term into the remote rather than having to type. That way, the interface stays simple even though complex input is possible.



**FIGURE 8-35**

Apple TV Remote, Remote App, and TV interface (images: Martin Charlier)



**FIGURE 8-36**

Amazon Fire TV with remote (image: Amazon)

## Universal Design and Accessibility

It is easy to forget that many products we work on as designers are inaccessible or much harder to use for people with disabilities like visual impairment. Sometimes products even become less accessible with the integration of new technologies (e.g., washing machines that move from physical controls to touchscreens). Connected products offer an opportunity to create more accessible and universal products.

In this section, we'll take a brief look at accessible and universal design in relation to connected products.

## ACCESSIBILITY

There are many aspects of a physical device that designers need to consider to make it appropriate for users with a certain disability. Checklists published by organizations such as the Royal National Institute of Blind People (RNIB) help designers with this task.

Here is an excerpt from the “handling” section of RNIB’s checklist “35 Questions to ask yourself when designing products for people with sight loss, disabilities and older people”:

CHECKLIST EXCERPT: HANDLING		
6. Is the product easy to orientate? Could a visually impaired and/or older person easily locate the front, back, top, and bottom of the product?	Yes	No
7. Does the product require little physical strength to use? Consider, for example, an older person with arthritis – would they be able to lift, open, turn, grip or rotate the product to use it effectively?	Yes	No
8. Does the product have smooth edges and surfaces (not sharp or rough) and is without finger traps? Consider that an older and/or visually impaired person may have limited vision, reduced reaction time, motor control and dexterity.	Yes	No

The full checklist is a great tool if you want to make your product more accessible. It is available here: [www.designingconnectedproducts.com/resources/35QuestionsChecklist.pdf](http://www.designingconnectedproducts.com/resources/35QuestionsChecklist.pdf).

Another simple but powerful method that helps designers make products accessible is to simulate disabilities. You can close your eyes and see if you can still use your prototype. You can even use props, often called “empathy tools” to help you. Wear earplugs and earmuffs to simulate deafness. Or wear gardening gloves to simulate how somebody with arthritis might experience your product.

Enabled by Design, a nonprofit organization hosting a community around the philosophy of “design for all,” uses such empathy tools in their workshops and hackathons (see Figure 8-37). Cofounder Denise Stephens, however, points out that empathy tools aren’t a replacement for working with real users that have lived with a disability. For that

reason, in addition to exercises using such tools, Enabled by Design’s hackathons bring together designers with “design partners,” people who live with disabilities that form teams with designers to collaborate.



**FIGURE 8-37**

At a Hackathon organized by the nonprofit “Enabled by Design,” participants used gloves and modified glasses to empathize with low dexterity or vision conditions (images: Enabled by Design and Matt Dexter)

A third strategy to create accessible products is to utilize existing accessible platforms. Android, iOS, and the Web all have great built-in accessibility features. If you don’t have the resources to make your device accessible, a less resource-intensive approach is to ensure you support these built-in features. You could make sure an app or website is available that lets users control every aspect of the product. Users can then use braille displays or built-in accessibility features like Apple’s VoiceOver to control your product.

### **UNIVERSAL DESIGN—WE’RE ALL DISABLED SOMETIMES**

It’s worth bearing in mind that designing for a particular disability doesn’t mean a product alienates or is less desirable for other user groups. In fact, it can mean quite the opposite. An example frequently used to illustrate this is the success of the OXO Good Grips brand.

Founded in 1990 on the philosophy of universal design, the company was at least partly inspired by the observation of the founder’s wife’s difficulties with ordinary kitchen tools: she was suffering from arthritis. But instead of staying a niche brand solving problems of users with specific needs, OXO Good Grips products became successful because they improved household products for anyone, not just specific groups.



Universal design is a term coined by architect Ronald L. Mace. Similar to “design for all,” it refers to a design approach that aims to make products and environments usable for as many people as possible, regardless of their age or ability. Rather than designing for certain disabilities, the approach aims to create a design solution that works for everybody, whether disabled or not. You could say the approach assumes that everybody is disabled sometimes: listening to music with headphones means you’re temporarily “deaf” to other audio alerts; carrying your grocery shopping in one hand means you temporarily only have one hand available for other interactions. Universal design aims to make products usable in the broadest range of situations, whether the user is permanently disabled or not.

When it comes to technology, this can mean including multiple modes of output and input. For example, including light output in a smoke alarm in addition to audio makes the product accessible to deaf users without alienating nondeaf users; or including audio cues and tactile signposting in a device like a washing machine makes it accessible to blind users without alienating the sighted.

Openness and compatibility with other systems also enable users to create such systems themselves. For example, the Nest smoke alarm and the LIFX connected light bulbs are compatible. This means users can set up light bulbs to flash red in case of an alert from the smoke alarm.

As part of your design process, you should spend some time examining your concept through the lens of universal design. The seven principles of universal design can help with that:<sup>15</sup>

*Principle 1: Equitable Use*

The design is useful and marketable to people with diverse abilities.

*Principle 2: Flexibility in Use*

The design accommodates a wide range of individual preferences and abilities.

---

<sup>15</sup> The seven principles of universal design were developed in 1997 by a working group of architects, product designers, engineers, and environmental design researchers.

### *Principle 3: Simple and Intuitive Use*

Use of the design is easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level.

### *Principle 4: Perceptible Information*

The design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities.

### *Principle 5: Tolerance for Error*

The design minimizes hazards and the adverse consequences of accidental or unintended actions.

### *Principle 6: Low Physical Effort*

The design can be used efficiently and comfortably and with a minimum of fatigue.

### *Principle 7: Size and Space for Approach and Use*

Appropriate size and space is provided for approach, reach, manipulation, and use regardless of user's body size, posture, or mobility.

## **Summary**

There are many different interaction channels through which users can control a connected product. They all have different benefits and drawbacks that designers need to be aware of to create the most appropriate interface.

New kinds of connected products might require hitherto unfamiliar combinations of inputs and outputs to create effective and user-friendly experiences. To identify these, designers shouldn't settle too quickly for familiar or established interfaces, whether they are common mobile UI patterns or the rows of LEDs and reset buttons used in many connected devices so far. Instead, they should experiment with and explore new possibilities of interacting with a device or system.

Deciding on the input and output capabilities of an embedded device almost always involves compromising on either the device's flexibility or its usability. It's the designer's role to pick the right trade-off. This decision is partly about product strategy, but also about human insight gained through thorough research and prototyping: What is the right sweet spot for the particular device that is being designed?

Using mobile apps or web UIs as main or part of the interface of a connected product has advantages and disadvantages. Developing for these platforms benefits from their popularity, the available resources, and the body of precedent projects. App platform features can be utilized and smartphone capabilities tapped into. However, the devices might not always be available, and going through a smartphone can be slower compared to direct on-device control.

As the number of devices people own increases, creating interfaces that require less attention, are less intrusive, and require less cognitive load will become more important. But only some interface types are suitable to create such glanceable or ambient interfaces.

When input and output capabilities are limited, designers need to develop strategies to enable edge-case interactions that require complex interaction. This can be about finding creative ways of using these limited capabilities, or about supporting other devices.

Designers need to make sure devices are accessible or universal. Supporting multiple ways to input and output, relying on accessible software platforms, and creating open systems that allow users to connect different devices together all help with this.

In summary, here are some of the things to consider with different interface types:

- *Physical controls* are great for direct, fast control, especially when precise or fine adjustments might be required. They also make it easier for users with impaired vision to still use a product. Bear in mind that physical controls can't be easily changed and you might have to keep labeling abstract or use screens.
- *Lights* can be useful to create glanceable and less intrusive information output, but using LEDs has limitations: you might need blink patterns and color coding to convey more complex information.
- *Screens* are of course useful to create dynamic interfaces and keeping the UI flexible and updateable, but having a screen often makes it difficult for design teams to keep interfaces simple because there is no extra cost to add more features or UI elements.

- *Audio output* often trumps light output and can grab the user's attention. It also inherently carries emotional qualities you need to design. On the flip side, it can quickly become annoying and users might want to turn it off.
- *Voice interfaces* are a powerful way to input or output fairly complex information and users can operate them while doing other things. At the moment, though, they tend to be unreliable, only available in particular geographic regions (due to language support), and often require an Internet connection for server-side processing.
- *Gestural input* works well for gaming, or when it is short and the commands are obvious. Longer interactions lead to fatigue and muscle pain, and false positive inputs can be an issue.
- *Tangible user interfaces* can be great for interactive experiences in museums, educational products, or for musical instruments. They are also interesting for giving digital content physical representations. Losing tokens or parts of the interface can render them unusable.
- *Tactile output* is a promising field with concept products conveying information in ways that are subtle and that require less attention. Vibration output aside, this is currently mostly research and there are few commercial products using tactile output.
- *Context-sensitive interfaces* can reduce complexity by making decisions for the user or tailoring interfaces based on context. They only ever sense data points that are proxies for context, and that might be wrongly interpreted. There is also a risk that users see them as patronizing or overbearing.
- *Computer vision and barcodes* can replace cumbersome input. But they can sometimes be a clumsy solution that is easier done differently.

## [Case Study 3]

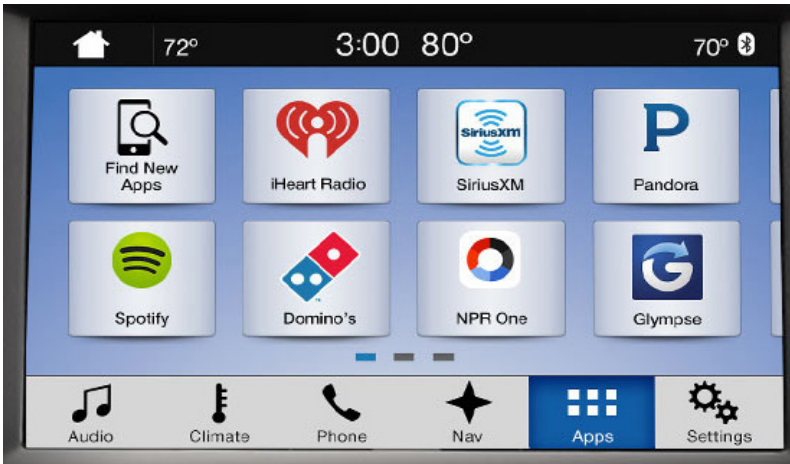
# Ford SYNC 3: Connected Car

BY PARRISH HANNA

## About Ford SYNC 3

“SYNC” is Ford’s brand for its in-car communications and entertainment connectivity system. The company currently has over 10 million SYNC-equipped vehicles on roads today across the planet. SYNC 3 is Ford’s third-generation voice-controlled connectivity system featuring an all-new hardware and software experience. More than 70% of all Ford customers in North America currently choose the touchscreen version of SYNC as an option within their new vehicle purchases. Multimodal input, connectivity, and system performance are the topics of the times in vehicle system enhancements, and this updated version is Ford’s latest upgrade in that race. This latest system provides simpler voice control as well as more intuitive touchscreen access to audio, phone, climate, navigation, and smartphone application functions.

Connectivity is now like the air that we breathe for many younger consumers. SYNC 3 introduces our third-party application interface, AppLink (see Figure CS3-1), which provides integration between SYNC and your connected smartphone. Using the SYNC touchscreen, users can interact directly with applications on the phone, such as Spotify, Glympse, Pandora, NPR News, and MLB.COM At Bat. SYNC 3’s software can be updated over-the-air via WiFi, or via USB. Clicking the “Apps” icon along the bottom row of the SYNC 3 touchscreen provides access to a vast array of third-party apps that have been optimized for use while driving. Apps for use via SYNC have to be redesigned to be safe while driving, while still maintaining the functional integrity of the app (and its brand). Available apps are easy to discover, as the tiles with each app logo appear on the touchscreen.



**FIGURE CS3-1**

Ford SYNC3 interface for AppLink

Our goal is always to design systems to enable the situations that people value—with a particular emphasis on clarity and ease of use, fast response, simplicity, and modernity. Multiple shapes, textures, tones, and behavior also need to be combined in such a way to communicate our intended Ford interactive brand experience. We need to support people's needs (by providing the technology infrastructure required), yet also direct their limited attention and prioritize their most important actions (by providing what we consider to be the optimal design for safe interaction).

Ford worked with Panasonic to develop SYNC 3. The software operating system is QNX running on a contemporary Texas Instruments multimedia processor. The navigation application is provided by TeleNav and map data is provided by HERE (formerly NAVTEQ). Our hardware and software decisions were driven by the need to satisfy and then exceed our consumer's expectations—particularly of performance and ease of use.

As vehicles rapidly evolve from simple mechanical machines to complex multimodal emotional platforms, what remains the same is the need to put the safety of all people first.

## Designing for Connected Car Systems

Like the operating system and digital ecosystem world of Apple, Google, Microsoft, and others, automobile manufacturers must now make strategic decisions regarding whether to develop open or closed systems, and the level of third-party integration and support they are going to provide.

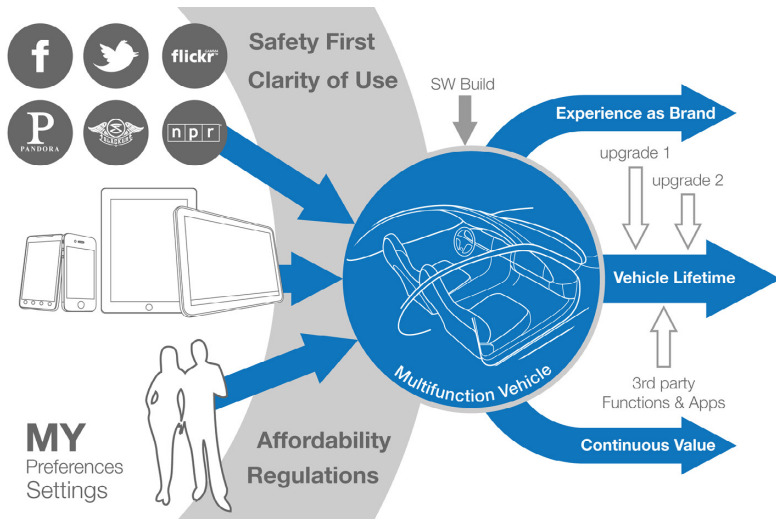
Regarding connectivity of your car, the latest options include supporting alternative OS integration from companies like Apple and Google, supporting a wide range of mobile device integration, and supporting mobile service and support providers like Verizon, AT&T, and others. In the future, “upgrades” may come from software companies and app developers, mobile phone manufacturers and service providers, and of course automotive companies. One potential outcome of this future of upgrades could be that vehicles adapt architectures of flexible components and hardware so that they can keep pace.

The addition of application programming interfaces (APIs) and a modem unleashes an unlimited world of functionality and content available in vehicles. The Digital Revolution is partially driving your car. When we consider new features, we have to decide whether it will be a built-in function (embedded in the system), or if it will be beamed in (delivered over the air) or brought in (transported via a personal phone or tablet). Complex enablers are continuously evolved to support each of those scenarios (see Figure CS3-2). New automobiles are high-speed multifunction devices.

### A Functional-Rich Small Space

The diversity of system integration in vehicles is vast and via connectivity and the evolution of sensors, is increasing at an accelerated rate. Across the domains of comfort, entertainment, and safety, we strive to satisfy basic needs, provide easy to use, clear systems, and spark surprise and delight through clever, thoughtful executions.

Our goal is to integrate multiple interactive systems in a layout and flow that is integrated, cohesive, and logical to the driver. And to do it in a way that is unique and recognizable to Ford or Lincoln.



**FIGURE CS3-2**

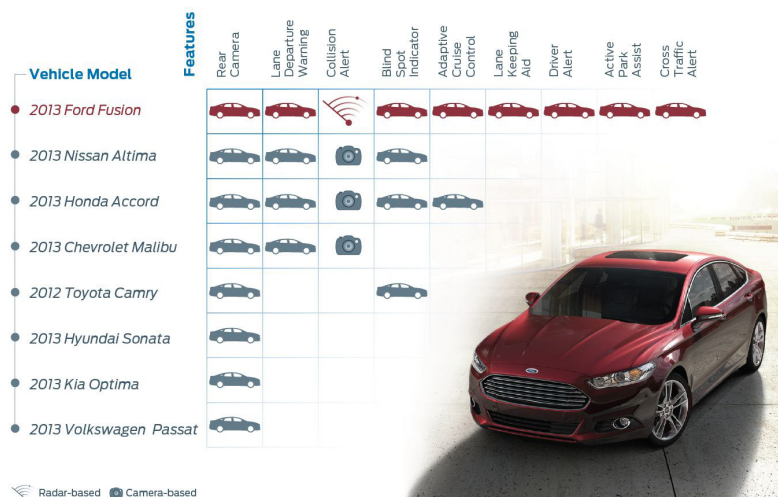
User experience filters in the connected car ecosystem

In the future, as our systems become more “smart,” perhaps they are more driver aware, contextually aware, situation aware, and they recommend and present functions as needed—or they interact on your behalf. Early examples of shifts from warning to intervention are present in industry features like Lane Keeping Aid, Active Park Assist, Collision Warning with Pedestrian Detection, and Traffic Jam Assist (see Figure CS3-3). When activated, each of these features will assist the driver with control of the vehicle, override the driver’s actions, and/or automate control of the vehicle in certain situations.

When designing the interior interactive experience, one of our biggest challenges is deciding the logical distribution and representation of content and functionality throughout a limited interior space. Should it be grouped by related functions, should it correlate to activities inside or outside of the vehicle, should it be grouped by proxemics and frequency of use/need? There are a limited amount of physical surfaces to continuously populate with physical buttons.

In theory, digital displays offer an unlimited amount of space (with pages, layers, 3D, etc.) to present features and functions. But navigating and interacting with these digital or virtual spaces while driving can be very challenging and unsafe. So, some design principles of physical space (like button size, readability, spacing, shape, etc.) are still vitally important in in-car digital interfaces.





**FIGURE CS3-3**  
Driver assistance features

## Like Your Tablet but Intentionally Different

Throughout the process of designing a touch interface for driving, we needed to translate traditional graphic design and interface design rules. These included the definition and design of optimized button sizes and behaviors as well as font characteristics that combine to maximize readability and interaction at a quick glance. The gesture vocabulary of your smartphone or tablet is not appropriate for touchscreens in cars. We studied the basic mechanics or motion required to perform a gesture along with the goal of the gesture or the intended task. Some relative gestures like pinch & zoom and swipe do translate, but others that require precise movements and control (like moving along a slider or drag and drop) were deemed as not appropriate.

Throughout the history of SYNC, voice input has continuously been evolved and improved to increase the safety of interaction. So, each new release supports enhanced voice features that better support the natural way people speak. Ease of user input and ensuring the system understands what users are saying is critical: particular priorities are addresses, locations, and other navigation-related items, as well as more personal content such as music, artists, and entertainment-related items. With SYNC 3, we have also integrated SIRI hands-free

capabilities. How many voice interaction systems will we associate with brands and will that association or recognition feel comfortable in other physical contexts?

Through observational research methods applied to iterative prototypes in both simulators and on the road, we learned how a vehicle touchscreen should differ from your personal device touchscreens (see Figure CS3-4). For instance, drivers did not see the value of a Homepage or the need for the vehicle to reset to a homepage with each key cycle. So, the next time that you turn on your vehicle, SYNC 3 retains the state that you left it in. Remember HyperCard?



**FIGURE CS3-4**

User experience testing of prototypes

We also came to thoroughly understand the most important and most frequently performed consumer tasks. The first of these, by far, is listening to and adjusting music. We also strive to easily get from point A to point B and when we want to be productive, we find ourselves making phone calls. So, we simplify and clarify the system design around the optimization of the best ways to complete these tasks. The SYNC 3 system also provides a display that can support other systems in the

car—like the cameras mounted outside the car, the multicontour seat adjustment, and other adjustments like volume adjustments and temperature changes.

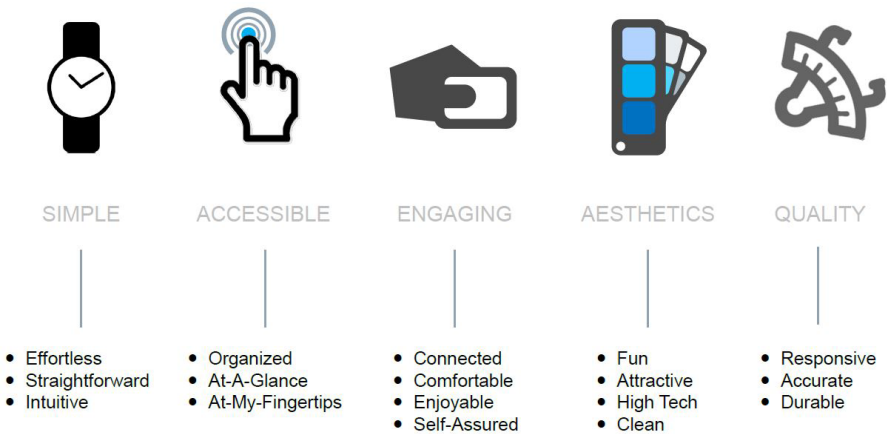
We also study the stereotypes or mental models of driving activities. When I ask drivers to adjust their seats, in most cases they move their hands around the bottom of the seat looking for adjustment buttons. When we ask drivers to play music, they used to look for a physical Play button, but now they look for it on the glass. When we ask drivers to adjust their cabin temperature, they tend to look for an adjustable knob or a physical Temperature Up button. However, we are seeing many high-end automotive companies move their climate controls to the glass. Is this a mental model that will change and become permanent?

The complexity of driving systems will continue to increase—particularly driven by a next wave of complex systems like driver assist technologies, driver recognition and personalization, computational learning systems, and ever more sophisticated brought-in devices. Over time, a continuous loop of learning ought to be built in to these systems. The integration and implementation of over-the-air (OTA) systems on the frontend, and data analytics capture and analysis on the backend are the platforms for continuous improvement. This “closed loop” of understanding and improvement is critical for success and safety. However, this level of infrastructure is costly and time consuming to achieve.

## **Demonstrating and Implementing an Iterative User-Centered Design Process**

As mentioned, SYNC 3 was designed and developed based on direct consumer feedback through market research and in-person clinics. All of Ford’s new product and service development is informed by the triangulation of different types of research, from lab based, to survey based, to direct observation of usage. Our previous SYNC systems have generated more than 22,000 data points, which were translated into many insights that were used to inform the design of the SYNC 3 system (see Figure CS3-5). Like carving a sculpture from a block of marble, we shaped the system through iterative prototyping—from low-fidelity paper and cardboard models to high-fidelity simulator and drivable prototypes—and continuous consumer validation and measurement.

Then statistical validation of large consumer sample sizes ensured that we were on the right track and had designed a highly competitive and useful system.



**FIGURE CS3-5**

Metrics developed from research insights

In general, we have noticed that drivers are self-aware and occasionally self-regulate and do not do a particular distracting activity in the car. But for the most part, they do them anyway (like eating, putting on makeup, or most frequently, texting) and their confidence artificially grows each time they complete the activity without a crash. And if an accident does occur, SYNC now can provide 911 operators with additional crash details, such as type of accident (front, side, rear, roll-over), change in speed, and number of safety belts detected, to help them better judge the severity of the accident and the emergency resources.

SYNC 3 will launch later this year in North America, first on 2016 model year products, and migrate to all vehicles in North America by the end of 2016 calendar year. After that, SYNC 3 will migrate into all of our vehicles globally.

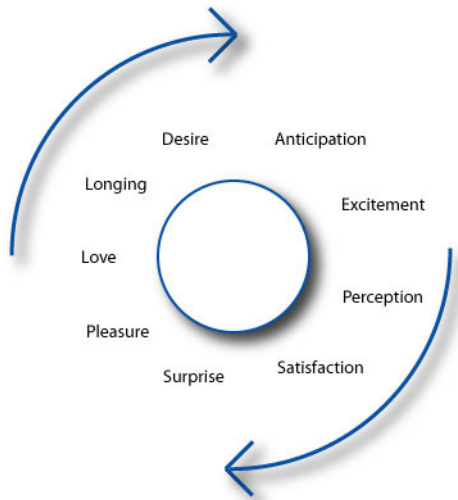
We are working with Apple on implementation of CarPlay in the future. We are working with Google on implementation of Android Auto in the future. Siri Eyes-Free allows drivers to initiate a Siri session in the same way as they do on the iPhone, but using SYNC's Push to Talk button on the steering wheel. Siri only will respond to commands with audio-based results, not visual images that require looking

at the iPhone screen. As product and service design cycles continue to shorten and new models of integration (like CarPlay and Android Auto) emerge, the possibilities of interaction and use will expand.

## From Designing Features to Designing for Experiences

The experiences that are afforded in today's automobiles are the result of the ad-hoc integration of disparate components and features, particularly in the realm of full digital displays, driver assistance, head-up displays, and ever more connected infotainment systems and touchscreens. In most cases, automotive OEM's do not know what the complete interior interactive system will be like until all of the interior components and software solutions are delivered from suppliers and bolted into the first drivable prototypes. By that time, specifications are most likely complete (or "locked") and changes and/or improvements are difficult to implement. Auto manufacturers must invest in iterative prototype development to learn and predict system performance and use.

In the future, Ford and Lincoln vehicles will be designed with the intent of enabling consumers to have specific experiences and form emotional connections to their vehicles and their related services and systems (see Figure CS3-6).



**FIGURE CS3-6**

The cycle of emotional design

## Mobility as an Emotional System

We also observe that consumers have strong emotional connections to their automobiles. They give them names, they refer to them as “an integral part of their group of friends,” and they think of them as an extension of themselves. The automobile originally enabled freedom on the open road, and it has evolved into much more. It now represents a place to stay connected to the world, your friends, and your family; to be entertained; to be productive; and to clear your mind and remove stress and anxiety from your life.

As a technology-infatuated society, we find ourselves addicted to being busy and that often involves being connected to digital technologies—we are addicted to screens. SYNC 3 was designed to appropriately fit the car, our ever-changing cultural and societal norms, and most importantly to fit our lives (see Figure CS3-7).



**FIGURE CS3-7**

The car is a now central part of the digital ecosystem

# Cross-Device Interactions and Interusability

**BY CLAIRE ROWLAND**

In systems where functionality and interactions are distributed across more than one device, it's not enough to design individual UIs in isolation. Designers need to create a coherent UX across *all* the devices with which the user interacts. That means thinking about how UIs work together to create a coherent understanding of the overall system, and how the user may move between using different devices.

This chapter explores *interusability*—the user experience of interconnected devices and cross-platform interactions—and how to make a bunch of diverse devices feel like they are working in concert.

This chapter introduces:

- Cross-platform UX and the need to design for systems, not just devices (see page 338)
- The concept of interusability (see page 340)
- The role of conceptual models in understanding what a system does, and why these are especially complex in IoT (see page 341)
- Composition: distributing functionality between devices (see page 350)
- Consistency across multiple UIs (see page 360)
- Continuity of data and interactions across devices (see page 368)
- Applying interusability thinking to broader contexts (see page 379)

This chapter addresses the following issues:

- What makes a cross-device system feel coherent (see page 338)

- Why it's complicated to understand how an IoT system works, and how we might help users with this (see page 343)
- Deciding how best to allocate roles and functions to different devices in the system (see page 357)
- Why terminology and platform conventions are especially important forms of consistency (see page 361)
- Dealing with data and content synchronization issues in the UI (see page 368)
- Designing interactions that require switching between devices (see page 376)

## Cross-Platform UX and Usability

Many of the tools of UX design and HCI originate from a time when an interaction was usually a single user using a single device. This was almost always a desktop computer, which they'd be using to complete a work-like task, giving it more or less their full attention.

The reality of our digital lives moved on from this long ago. Many of us own multiple Internet-capable devices such as smartphones, tablets, and connected TVs, used for leisure as well as work. They have different form factors, may be used in different contexts and some of them come with specific sensing capabilities, such as mobile location.

Cross-platform UX is an area of huge interest to the practitioner community. But academic researchers have given little attention to defining the properties of good cross-platform UX. This has left a gap between practice and theory that needs addressing.

In industry practice, cross-platform UX has often proceeded device by device. Designers begin with a key reference device and subsequent interfaces are treated as adaptations. In the early days of smartphones, this reference device was often the desktop. In recent years, the “mobile first” approach<sup>1</sup> has encouraged us to start with mobile web or apps as a way to focus on optimizing key functionality and minimize “featuritis.” Such services usually have overarching design guidelines span-

---

1 See Luke Wroblewski's *Mobile First* (A Book Apart LLC, 2011), <http://abookapart.com/products/mobile-first>.



ning all platforms to ensure a degree of consistency. The aim is usually to make the different interfaces feel like a family, rather than on how devices work together as a system.

This works when each device is delivering broadly the same functionality. Evernote, eBay, and Dropbox (see Figure 9-1) are typical examples: each offers more or less the same features via a responsive website and smartphone apps. The design is optimized for each device, but provides the same basic service functionality (bar a few admin functions that may only be available on the desktop).



**FIGURE 9-1**

Evernote offers broadly the same service functionality across different device types (image: Evernote)

But this approach breaks down when the system involves very diverse devices with different capabilities working in concert. In IoT, many devices do not even have screens, or an on-device user interface. Multiple devices may have UIs with very different forms or specialized functionality (see Figure 9-2). Even if the UI is only on one device, the service still depends on all the devices working together in concert.



**FIGURE 9-2**

The SmartThings ecosystem contains a range of specialized devices that complement each other (image: SmartThings)

It's not possible to design a system like this by thinking about one device at a time: this is likely to create a disjointed experience.

In order to use it effectively, the user has to form a coherent mental image of the overall system. This includes its various parts, what each does and how different objectives can be achieved using the system as a whole. Traditional single-device usability doesn't tell us very much about how to do this.

## What Is Interusability?

Charles Denis and Laurent Karsenty first coined the term “inter-usability” in 2004 to describe UX across multiple devices.<sup>2</sup> Conventional usability theory is under-equipped to cope with cross-platform design. However, one 2010 paper by Minna Wäljas, Katarina Segerstahl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen proposes a practical model of interusability.<sup>3</sup>

Wäljas et al. propose that the ultimate goal of cross-platform design is that the experience should feel *coherent*. Does the service feel like the devices are working in concert, or does the UX feel fragmented?

They define three key concepts for cross-platform service UX, which together ensure a coherent experience:

- *Composition* (how devices and functionality are organized)
- Appropriate *consistency* of interfaces across different devices
- *Continuity* of content and data to ensure smooth transitions between platforms

---

2 C. Denis and L. Karsenty, “Inter-Usability of Multi-Device Systems—A Conceptual Framework,” in *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, eds. A. Seffah and H. Javahery (Hoboken, NJ: Wiley).

3 M. Wäljas, K. Segerstahl, K. Väänänen-Vainio-Mattila, and H. Oinas-Kukkonen, “Cross-Platform Service User Experience: A Field Study and an Initial Framework,” *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI 2010*, p. 219. ACM, New York (2010). Paper available at <http://bugi.oulu.fi/~ksegerst/publications/p219-waljas.pdf>. I'll refer to this paper several times in the rest of this chapter as Wäljas et al., although I understand that Katarina Segerstahl was the primary researcher. Her PhD, available at <http://herkules.oulu.fi/isbn9789514297274/isbn9789514297274.pdf>, builds on the same concepts.

The paper was published in 2010 and the services evaluated (including Nike+ and Nokia Sportstracker) now inevitably feel a little dated. But we have found the model still holds up well in our own work designing IoT services, and it's a key reference for the rest of this chapter.

## Conceptual Models and Composition

In this section, we'll look at two related concepts that help us design systems spanning multiple devices.

*Conceptual models* refer to the way humans understand the overall system (and its interfaces) to work. Users need some understanding of how the system works in order to figure out how to interact with it. As we just saw, *composition* is a dimension of interusability. It refers to the way user-facing functionality is distributed between different devices: which device does what. The two concepts are related in cross-platform design: understanding which device does what is part of forming an effective conceptual model.

### CONCEPTUAL MODELS

#### The user model and the design model

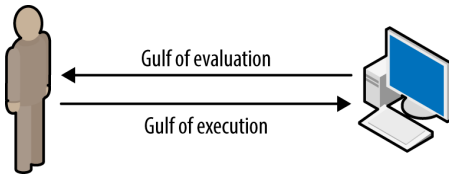
The conceptual model may refer to the way the user understands the system, or the way the designers (or engineers) think about the system. Users develop a mental model of the system (a *user model*) that enables them to understand what it does, how to interact with it, and how it will behave. At first, this will be based on prior experience of other systems or similar activities. Over time, they will develop the model through their experiences with the system itself. The way the designers or engineers think about the system will be reflected in the *design model* (this distinction was defined in Don Norman's *The Design of Everyday Things*).<sup>4</sup>

As Norman puts it: “The problem is to design the system so that, first, it follows a consistent, coherent conceptualization—a design model—and second, so that the user can develop a mental model of the system—a user model—consistent with the design model.”

---

4 Donald Norman, *The Design of Everyday Things* (New York: Basic Books, 1988).

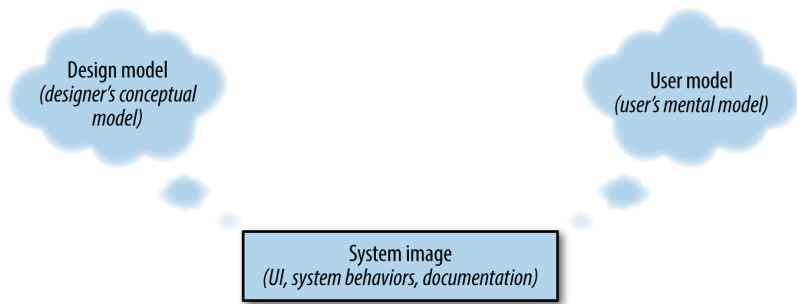
The similarity between the design model and the user’s mental model is a core determinant of usability in any system, not just IoT. How easy is it for the user to figure out how to achieve a particular goal using the system (which Norman refers to as bridging “the gulf of execution”<sup>5</sup>)? How easy is it for the user to understand what the system does in response (“the gulf of evaluation”)? (See Figure 9-3.)



**FIGURE 9-3**

The gulfs of evaluation and execution

Ideally, the user model maps closely onto the design model. But frequently, the design model may not be a good fit for what the user wants to do, or users may only partially understand it. If the system doesn’t conform to any prior expectations, users must develop a new mental model, based on trying to infer the design model. Users learn about the design model through the interface, behaviors of the system, and documentation—which Norman refers to as the *system image* (see Figure 9-4).

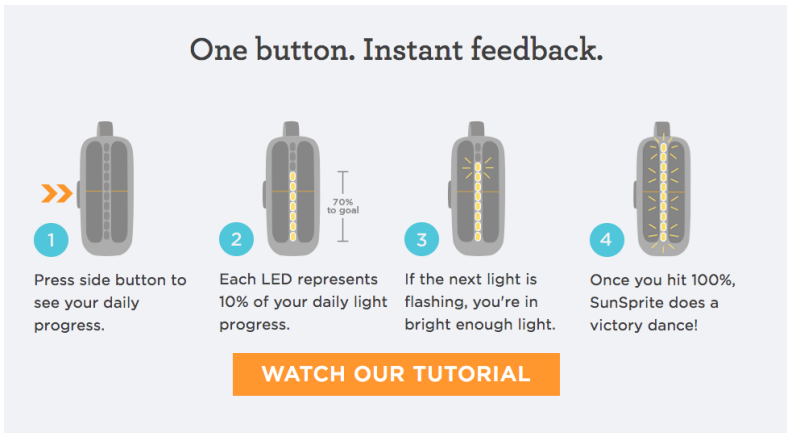


**FIGURE 9-4**

Diagram: the user, system, and design model (redrawn from [http://www.jnd.org/dn.mss/design\\_as\\_communication.html](http://www.jnd.org/dn.mss/design_as_communication.html))

5 Donald Norman, “Cognitive Engineering,” in *User Centered System Design: New Perspectives on Human-Computer Interaction*, ed. Norman and Draper (Hillsdale, NJ: Lawrence Erlbaum Associates, 1986).

You can't design the user's mental model directly. But you can design the system image to convey the design model clearly (see Figure 9-5). You should also define your design model explicitly; to make sure it's clear, consistent, and not overly technical or complex for your audience.



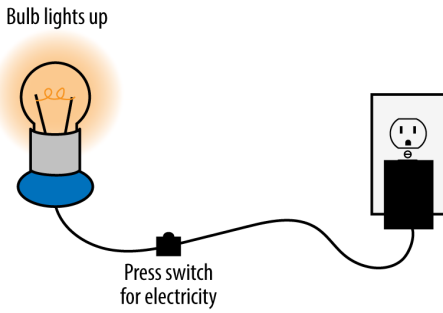
**FIGURE 9-5**

A system image: the SunSprite Tracklight helps users monitor their daylight exposure; the instructions explain what it does, how to interact, and how data is displayed using the LEDs (image: SunSprite)

### **Multidevice services are conceptually more complex**

Back when Norman first wrote about conceptual models, a system was generally a software application running on a standalone computer. Multidevice services make conceptual models more complicated. There are not just more interfaces, but more places where processing and functionality can live and where data can be stored. Because there are more nodes and connections, there are more points of failure and ways to fail. This is often where complexity is exposed: when the system is working well, it may not matter where your data or preferences are stored. But when parts or connections fail, the user has to understand something about how the system works in order to understand what is happening and why.

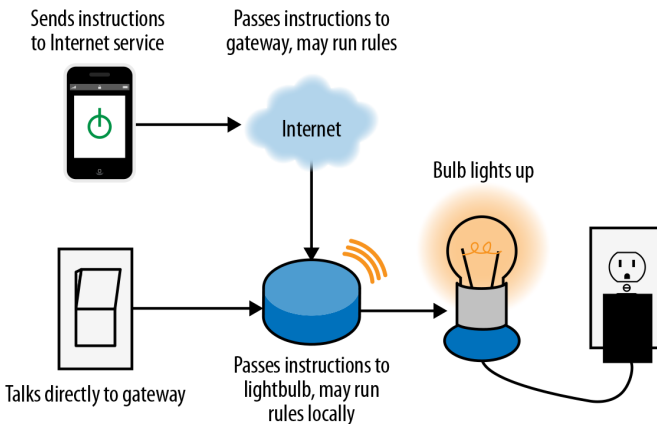
Take the example of a lighting system. The mental model of a lamp is simple: it has power, a switch, a fitting, and a bulb (see Figure 9-6). If the lamp doesn't work, it's probably because the power has failed or the bulb has blown.



**FIGURE 9-6**

A conventional lamp has a simple conceptual model

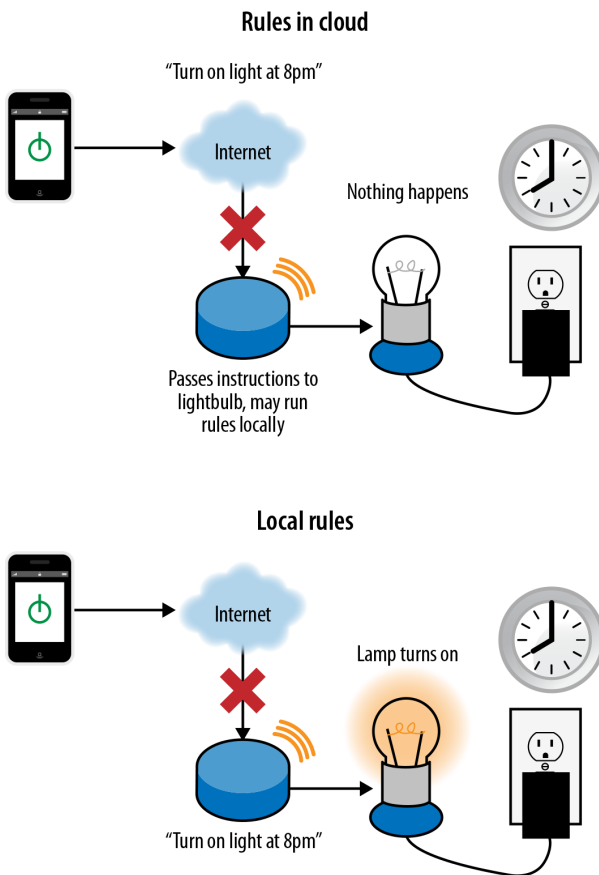
A connected lighting system has bulbs, switches, fittings, and power too (typically either the bulbs or switches will be connected). It also has an Internet service, probably hosted remotely. It has a smartphone app and perhaps a web app, too. It probably also has a gateway device. It has more parts (see Figure 9-7) and more different kinds of part. It can also do more. It may run automated rules to turn lights on and off at certain times, or when certain trigger events happen (such as the security alarm being activated). The intelligence that controls the system may live in several places: in the bulb or switch itself, in the gateway, in the Internet service, or even in the smartphone app.



**FIGURE 9-7**

Connected lighting can do more than conventional lighting, but the conceptual model is more complex

When everything is working well and connected, users don't need to concern themselves with which code is running where. But if a part of the system develops a fault or loses connectivity or the network is slow, the impact will depend on what that device is doing. Will the lights (or intruder alarm, or heating) stop working if the user's phone battery runs out or they have no signal? Or will they keep running even if the user cannot remotely access the home at that point? What if the home Internet connection goes down? Will lighting rules continue to work locally? If they are stored in the gateway or edge device, they will. If they are stored in the Internet service or smartphone, they will not (see Figure 9-8).



**FIGURE 9-8**

Home automation routines stored in the cloud will not run if the Internet connection goes down; if they are stored locally, they will continue to run, but the user won't be able to see this or control devices remotely

How you choose to distribute system intelligence is a system architecture issue, as discussed in Chapter 3. What's most appropriate for your system will depend on what it does and your users' expectations. An intruder alarm should not fail completely because the Internet went down. But it's not a disaster if your energy monitoring system is occasionally unavailable for short periods of time, as long as data is not lost.

The new challenge for UX is that this is a lot of complexity that users didn't previously have to worry about (see the discussion of the "surprise package" in Chapter 4). There are two ways to deal with this complexity: you can explain it, or try to hide it.

Although BERG is now defunct, the BERG Cloud bridge, the gateway for Little Printer (see Chapter 7), was a simple example of a device that explained how the system was working. It had LEDs to show whether the device had power and an Ethernet connection, and upstream and downstream connectivity (see Figure 9-9). It was labeled to explain that upstream meant that the bridge could see the BERG cloud Internet service, and downstream meant that the ZigBee network used to connect to local devices was running. The gateway was communicating the system image.



**FIGURE 9-9**

The BERG Cloud Bridge (image: BERG)

The more complex the system, the more overwhelming it may be to explain in detail. In that case, it would be better to allow the user to work from a simplified mental model. Automatic gearboxes are complex mechanical systems that require only a simplified mental model



in order to use (see Figure 9-10). But this is a hard trick to pull off. The gearbox is mature technology and only performs one basic function. Also, most consumers can draw from the experience of driving cars. IoT technology is newer, and often does things that are less familiar to users. Many IoT systems also have multiple functions, so it can be harder to reduce them to a simplified conceptual model.



**FIGURE 9-10**

Automatic gearbox controls hide the complexity of the system behind a simplified conceptual model

One example of a product with a simplified conceptual model is Apple iBeacons (beacons were discussed in Chapters 2, 3, and 4). Imagine a user walks into a store for which they have an app installed, their location is detected using iBeacons, and they are sent a push notification about a special offer. It's good enough for that user to understand that the store has sent them a message because they are physically there. They might well think that the beacons are beaming them the message (news articles even describe beacons as working in this way).<sup>6</sup>

That model isn't accurate. What actually happens is: the beacon broadcasts a data packet containing a unique iD and information about where it is based (e.g., the store). When the iOS device detects the broadcast,

---

<sup>6</sup> <http://bit.ly/t15VBzF>

it wakes up the relevant app (e.g., the store’s loyalty app). That, in turn, triggers the app to display a notification: the app might check into the backend Internet service in order to see what notifications are available. The beacon isn’t sending the notification, it’s just acting as a tripwire that tells the store’s app (and backend service) where the user is.

That’s a lot more complicated. If the store tried to explain that to their customers, they’d probably just confuse them. In this situation, a simplified conceptual model is good enough to use the system perfectly well, even if it leads the user to some wrong assumptions.

Note that in this case, there’s nothing that can go wrong as a result of the user not understanding the system model. In other situations, a misunderstanding might have serious consequences. For example, if activating an emergency alarm, the user does not just need to know that pressing the button summons help. They may need to know whether pressing the button is guaranteed to get the message through, and whether they should wait for confirmation that someone has received it.

UX researchers at Ericsson have suggested that it is particularly difficult for users to understand *networks* of devices. Their informal research indicated that users currently think of connections between devices as being “invisible wires.”<sup>7</sup> As Ann Light, coauthor of Chapter 11, puts it: “Most people are disposed to think of things, not links; of nodes rather than relations.”<sup>8</sup> But this way of understanding is not helpful in making sense of complex networks with many interconnections and interdependencies. *To understand a system, users must understand the links as well as the nodes.*

In 1983, the HCI specialist Larry Tesler (then at Apple) proposed the Law of Conservation of Complexity. Interviewed in Dan Saffer’s book *Designing Interactions*, Tesler says: “I postulated that every application must have an inherent amount of irreducible complexity. The only

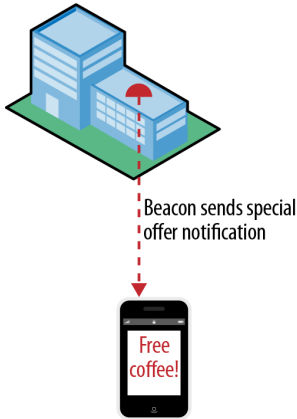
---

7 Joakim Formo, “The Internet of Things for Mere Mortals,” <http://www.ericsson.com/uxblog/2012/04/the-internet-of-things-for-mere-mortals/>.

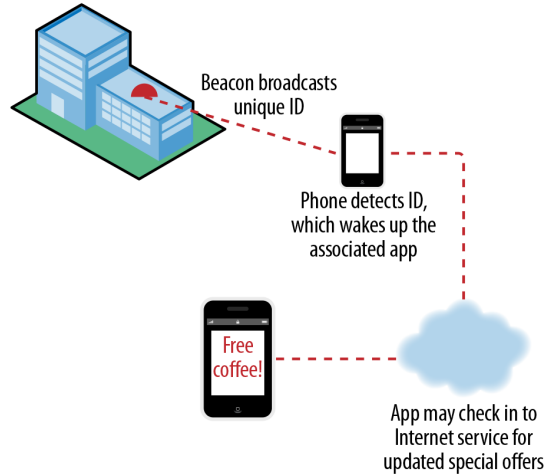
8 In conversation.

question is who will have to deal with it.”<sup>9</sup> His point was that shielding users from complexity would involve extra work from designers and developers.

### What the user needs to know



### What's actually happening



**FIGURE 9-11**

iBeacons diagram: what the user needs to know, and what's actually happening

User understanding will improve over time with familiarity, but only if we, as designers, help them with clear system images. We need to figure out what complexity users will need to deal with, and where products and tools can be simplified. As a general rule, if the task or activity the user wants to perform is complex or requires a high level of skill, it's appropriate for the user to engage with that complexity. Or perhaps it's a job for a professional. If the task or activity can be expressed simply but the technology is complicated, there's a good case for designing around a simplified mental model.

We don't yet know what that looks like for IoT. For starters, if you need to explain to the user that part of the system is not working, it's important to explain why and what this means. For example, if you are alerting them that the security alarm has lost Internet connectivity, you

<sup>9</sup> Dan Saffer, *Designing for Interaction* (San Francisco: New Riders, 2006). Larry Tesler interview available at <http://www.designingforinteraction.com/tesler.html>.

might choose to tell them that the cameras and alarm sounder are still active but that they will not receive alerts. Or if the user is traveling in a different time zone, you might want to show the current time at home on the heating control app, to indicate that schedule changes are based on the local time zone of the controller, not where they are now.

## COMPOSITION

Composition refers to the way the functionality of a service—especially the user-facing functionality—is distributed across devices.

Good composition distributes functionality between devices to make the most of the capabilities of each device. Designers should take into account the context in which each device will be used, and what users expect each to do.

### Patterns of composition

There are some common patterns to composition. Web services delivered across smartphones, desktops, tablets, and connected TVs are often *multichannel*. Each device provides the same, or very similar, functionality (in other words, there is a high level of *redundancy* between devices. Kindle, Netflix (see Figure 9-12), BBC iPlayer, Facebook, and eBay are all examples. Devices with small screens or limited input capabilities may only provide a subset of key functionality. But each device offers a similar basic experience of the service. Many users won't own or use all the possible devices on which the service could be used and this doesn't matter. It's perfectly possible to use the service via a single device and still have a good experience.

However, many IoT services run on a mix of devices with different capabilities. Service functionality and user interactions will be distributed across different devices. Some functionality may be exclusive to a specific device. For example, in the Withings ecosystem, only the scale can measure body mass and only the blood pressure monitor can measure blood pressure. Some devices may be custom designed for the service. We've seen many examples of these already throughout the book, from connected door locks to smart watches and thermostats.



**FIGURE 9-12**  
Netflix is a multichannel service (image: Netflix)

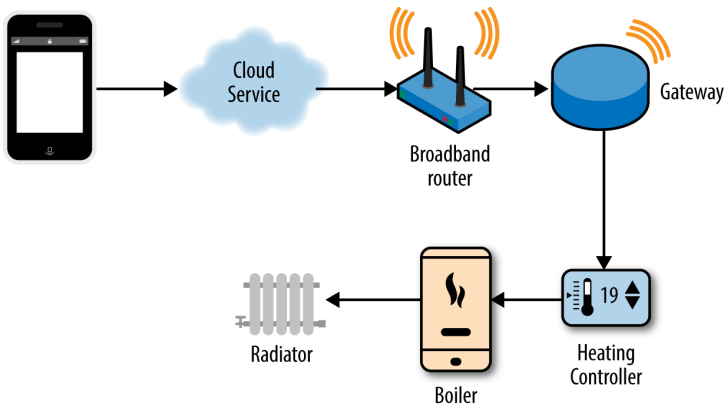
For reasons of cost, or practicality, these may have limited inputs and outputs that are quite different from a conventional “computer” UI. A door lock may have a keypad and handle and perhaps an LED to show whether it is connected or not. It probably doesn’t have a screen. In these systems, we have to figure out which device handles which functionality. Each device (and the Internet service itself) may have a different role in terms of providing user interactions, connectivity, information gathering, processing, or display. In the terminology of Wäljas et al., this is a *cross-media* system (Figure 9-13 is an example).



**FIGURE 9-13**  
Withings is a cross-media ecosystem (images: Withings)

For example, a heating service as shown in Figure 9-14 may comprise:

- A boiler that heats the water
- An in-home controller/thermostat that tells the boiler when to switch on or off (this may be a separate thermostat and programmer, or just one device)
- A gateway that provides a low power connection to the controller and bridge out to the Internet
- A cloud service that stores user account information and remote access to the system
- Smartphone and web apps that connect to the cloud service



**FIGURE 9-14**  
System diagram of a heating system

Both the Tado and British Gas Hive systems work in this way, but user-facing functionality is distributed differently. The Tado thermostat/heating controller has almost no UI (see Figure 9-15). Users can view the current temperature and set or alter the setpoint, but most interactions are handled on the smartphone.<sup>10</sup> This may keep manufacturing costs down. Smartphone interfaces are much cheaper to develop than physical interfaces, as components like screens and buttons are relatively expensive. It's an elegant choice for a small household

<sup>10</sup> The first-generation Tado controller had no onboard controls at all: all interactions were via the smartphone.

occupied mainly by smartphone owners, but there are trade-offs. If you don't have your phone in hand, or the battery is dead, or you're a guest in the house without access to the phone UI, you have limited control.



**FIGURE 9-15**

The Tado heating controller and smartphone app (image: Tado)

The Hive Active Thermostat heating controller is a standard thermostat with on-device controls (see Figure 9-16). The device is designed to be competitive on cost with nonconnected heating controllers. So interaction design has to work within the constraints of an LCD screen and limited number of buttons. However, heating can also be controlled by phone and web apps, which are probably easier for most people to use than the hardware. This means that heating controls are available to anyone in the house, whether they have access to the smartphone app or not.

Some devices may not support any user interactions at all (see Figure 9-17). Some devices may be simple sensors that simply provide data to the service, as in an air quality monitoring system. In this case, you may simply hand off all functionality onto a single mobile or web app. Although the overall service may be complex, the web or smartphone UI in this case is in some ways simpler to design, as there is only one interface to consider. (Chapter 8 considers the pros and cons of handling functionality via a mobile device versus a specialized embedded device).



**FIGURE 9-16**

The Hive Active Thermostat heating controller and smartphone app (images: British Gas)

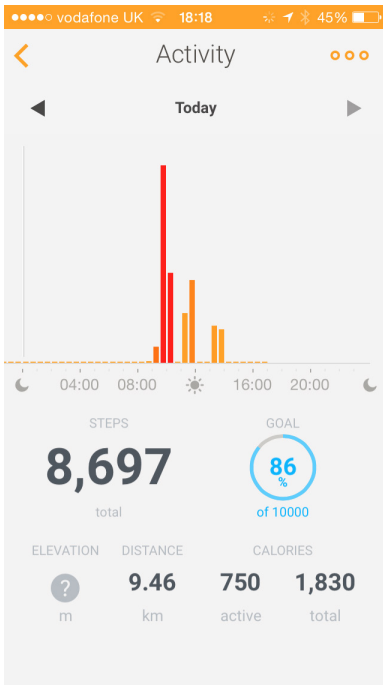


**FIGURE 9-17**

The Bluespray garden irrigation controller is entirely controlled by a smartphone app (image: Bluespray)



When key tasks are available across multiple devices, users may still be able to use the service even when some devices are unavailable. For example, the Withings smartphone app can use the onboard accelerometer to measure activity, so even if the user has forgotten their dedicated activity monitor, they need not lose data (see Figure 9-18).



**FIGURE 9-18**

The Withings mobile app can use the phone's onboard accelerometer to measure activity

Even where different devices support the same tasks, they may be used in different situations. For example, the key advantage of connected heating systems is that the smartphone app enables control from anywhere, whether that's the other side of the world or the user's bed.

However, this isn't a recommendation to duplicate every piece of functionality across every device—redundancy isn't necessarily a good thing. Too many functions on a single device can make the UI harder to use, especially if the device has limited input/output capabilities. And user interaction components, such as screens and buttons, add

significantly to manufacturing costs of embedded devices. The right decision will balance the usefulness, cost, and usability of putting various features on different devices.

For many systems, it makes sense to use a network of devices that are specialized for particular functions. For example, the Lively elderly care service uses specialized sensors to monitor the pillbox, fridge, and the kitchen (see Figure 9-19). This is referred to as *synergistic specificity*:<sup>11</sup> specialized components working together to deliver a service that is more than the sum of those components.



**FIGURE 9-19**

Lively elderly care system: safety watch for summoning help, with pedometer and medication reminders, hub, sensors for fridge and pillboxes, and a custom sensor containing an accelerometer to detect movement (image: Lively)

Users may also want to add (or remove) devices to suit their individual needs, or combine them in different ways to fulfill different purposes—this is a *modular* system. In some cases, different devices can be used to perform different functions as part of different services. For example, a home monitoring system may offer contact, temperature, moisture, smoke, and motion sensors. These could be used to detect occupancy for heating, lighting, and potential safety problems or intruders. A highly modular system can be very powerful. It fits well

---

11 M. A. Schilling, "Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity," *Academy of Management Review*, 25 (2000).

with the Web philosophy of “small pieces, loosely joined” (as described in Chapter 3): specialized components designed to be flexibly combined for many uses.

But systems of multiple devices will often be more complex for users to understand, configure, and use. To use the distinction from Chapter 4, it’s likely to be more of a tool for early adopters than a product that the majority of consumers will configure themselves. It’s also more complex for designers to communicate which devices are doing what at each point.

**TABLE 9-1.** Two key composition patterns (many systems are somewhere in between, with some specialized devices and some functionality available on multiple devices)

MULTICHANNEL SYSTEM	CROSS-MEDIA SYSTEM
Same service functionality available across multiple devices	Specialized devices with different capabilities provide a service that is more than the sum of its components

### Determining the right composition

For any service, there is often more than one possible suite of devices that could be used to deliver the service. The decision as to which is most practical will be influenced by the following factors. You may wish to prioritize, per task, which devices are optimal for each task, acceptable for the task, or not possible for this task.

**What best fits the context of use?** The first consideration is what best fits the activity, situation, and user needs. Certain devices need to live in one place where only one function is required (e.g., blind/window shade controllers or light switches). Others are used for activities, or in environmental conditions, that place constraints on form factors and interaction modalities. For example, climbers need their hands free and are out in the open. Delivering altitude, weather, and location information to a weatherproofed wrist-top device makes more sense than using a smartphone. Using a mobile phone’s music controls while driving would be dangerous. Key functionality should be mirrored on the car dashboard in a way that minimizes demands on attention. Features that are essential to one device may be inessential or even inappropriate for others.

**What connectivity and power issues do you need to consider?** The availability and reliability of the network connection is also key. If you have a good reliable connection, you can afford to centralize more functionality (e.g., putting your irrigation system controls on a smartphone).

It is simpler to handle information processing in the cloud, as with the Withings scales and fitness trackers, which simply take readings, display them in real time, but handle all other functionality in the online service. A weight and fitness service can handle temporary losses of connectivity gracefully, by storing data locally and syncing when the connection is available again.

However, for other types of service, this will not be acceptable (e.g., where safety or security is at stake). If a monitoring system for an elderly person loses connectivity, it might be acceptable for motion sensor data to be temporarily unavailable to the care giver, as long as it is clear that connectivity has been lost and live data is not available. But it would be completely unacceptable for the elderly person to be unable to use their emergency alarm during this time: the alarm should be able to fall back to using another form of connectivity.

In other words, the designer has to make an informed call on which tasks need to be available in different conditions: offline? With no power? Does the system make no sense if connectivity is lost? Is a suitable fallback available? I should not have to worry about being unable to enter and leave my own house because the front door lock has lost connectivity or has no power—and many other household functions must also be taken for granted to be effective. If you can't afford to lose access to functionality, you may need user controls (and perhaps more onboard intelligence) in the edge devices.

**Can you work with preexisting devices?** What hardware can you assume your user base already has and is familiar with using? For example, if users all have smartphones you can use these to handle complex interactions, determine location and identity, and in some cases handle local and Internet connectivity.

Smartphones come with onboard sensors (such as accelerometers), which can be used for tasks such as activity tracking without additional hardware. However, custom form factors (such as wristbands) may provide a better experience in some contexts of use, such as the forthcoming clip-on fall sensor for the Lively safety watch for older

adults. Specialist equipment may also tend to offer better performance through better quality parts, such as more powerful GPS chips or better battery life.

**What interaction capabilities do the various devices have (or could you cost-effectively include on a custom device?)** You may be able to consider adding or removing interaction capabilities (like screens, buttons, audio beeps, or LEDs) to embedded devices. However, these typically add to production costs, so you will probably need to keep these to a minimum and offload more complex functionality onto a mobile or web UI. If you're not able to influence the design of the embedded devices, you'll have to work with the interaction capabilities you have.

You may also decide that just because a device *could* support a particular function, it does not have to. Keeping things simple may make the device interface easier to understand. For example, a heating controller with a low-resolution screen and limited buttons might be best used for status information and in-the-moment controls (turn the heating up now!). You could offload more complex tasks such as schedule setting onto a web or mobile interface. The bigger screen size and richer interaction capabilities will enable a better design. You can also provide a “good” way to do the task on a fuller featured device and a limited or compromised version on a less capable device that must occasionally work alone. For example, an intruder alarm system may provide an easy way to view which sensors triggered the alarm on a mobile interface. The task may also need to be possible on the alarm panel via a basic LCD screen, even though this is likely to involve many more button presses, perhaps navigating menus and modal states.

**Does the system need to work if some devices are unavailable?** What happens if a device is unavailable (e.g., a smartphone is lost or the battery is dead)? Does it need to be used by third parties who may not have access to a web or mobile app, such as visitors to the home? Can the device work offline?

**How accurate does sensing need to be?** If a service needs to know the rough location of its user, a smartphone can estimate this from for GPS/cell tower signals. If it needs to know which room the user is in at

home (e.g., to turn the lights on and off), a smartphone could be used via Bluetooth LE connections, but will only be accurate if the user carries it with them at all times.

**Do users have set expectations of devices?** Users may expect certain devices to conform to familiar form factors, or provide familiar functionality. For example, they are likely to expect a heating controller to have some way of turning the heating on, or up.

**How do you balance cost, upgradeability, and flexibility?** User interface components, such as screens and buttons, are expensive to add to embedded devices. You may therefore decide to limit interactions on the embedded devices themselves and do most of the interaction “heavy lifting” via mobile apps or web interfaces.

It can be difficult to add new features to devices that are already out in the field, especially if this requires modifications to the interface. Again, offloading interactions to smartphone and web apps, which can be modified relatively cheaply and quickly, may make sense.

**How central to the service are the devices?** Are devices central to the conceptual model, or not? This may not affect the distribution of functionality, but it will affect the way in which you communicate the composition of the system.

## Consistency

*Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.*

**JAKOB NIELSEN, 1994**<sup>12</sup>

Consistency is well known as a general UI design heuristic. It’s a simple concept to grasp. But deciding what needs to be consistent, and what does not, can be tricky. You may have to trade off one type of consistency for another. Do you make all the buttons look the same so they are easy to identify as buttons? Or does that cause confusion by implying that certain functions are similar, when in fact they are not? Too much consistency, or consistency between the wrong things, can be as damaging as too little.

---

<sup>12</sup> J. Nielsen, “Heuristic Evaluation,” in *Usability Inspection Methods*, eds. J. Nielsen and R.L. Mack (Hoboken, NJ: Wiley).

In the case of cross-platform systems, designers also need to consider consistency across different devices. Consistency works to create a sense of coherence of the overall system.

Words, data, and actions that are the same across devices should be understood to be the same. Words, data, and actions that are different should be understood to be different. This helps users form a clear mental model of the system and its capabilities. Knowledge that users have gained about the system from one device can be transferred to help them learn how to use other devices.

Other elements that may need to be consistent to some degree across devices include the following:

- Aesthetic/visual design (to make the devices look, feel, and sound like a family)
- Interaction architecture (how functionality is organized)
- Interaction logic (how tasks are structured or the types of control used)

## **GUIDELINES FOR CONSISTENCY ACROSS MULTIPLE DEVICES**

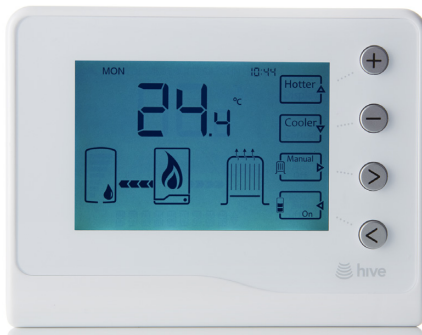
Determining what should, and what should not, be consistent across multiple devices is also a question of trade-offs. Should you try to make your UIs as consistent as possible across all devices in the system, or should you try to follow the conventions that may exist for individual platforms (such as mobile operating systems)? In this section we propose a set of guidelines to help you balance those decisions.

### **Use consistent terminology**

As a rule of thumb, the highest priority is to use consistent wording across devices. This ensures that data and actions across different platforms are understood to be the same thing. Whatever the display capabilities of each device, you can always give functions or data the same label even if you can't make them look the same.

For example, imagine you are working on a connected heating controller that offers three mode options: ON (heating is on continuously), AUTO (heating is running to a preprogrammed schedule), and OFF (see Figure 9-20). These are already set in the fixed segment LCD display and cannot be changed. You might think that ON and AUTO are not the clearest terms. You'd prefer to change them to CONTINUOUS

and TIMER or SCHEDULE in the smartphone app interface. However, this would create a disconnect: users then have to understand that ON and CONTINUOUS are the same thing.



**FIGURE 9-20**

The Hive Active Heating controller is an example of a device with fixed segment LCD labels mapped to physical buttons (image: British Gas)

Users are often intimidated by heating controllers and expect them to be confusing. And they might not have a strong enough mental model of the system to infer that CONTINUOUS and ON are the same thing. Having tested systems with similar issues, we found it was more important that these options were consistently named across devices. The value of better terminology in the mobile app is undermined if users don't understand that the functions are the same.

### **Follow platform conventions**

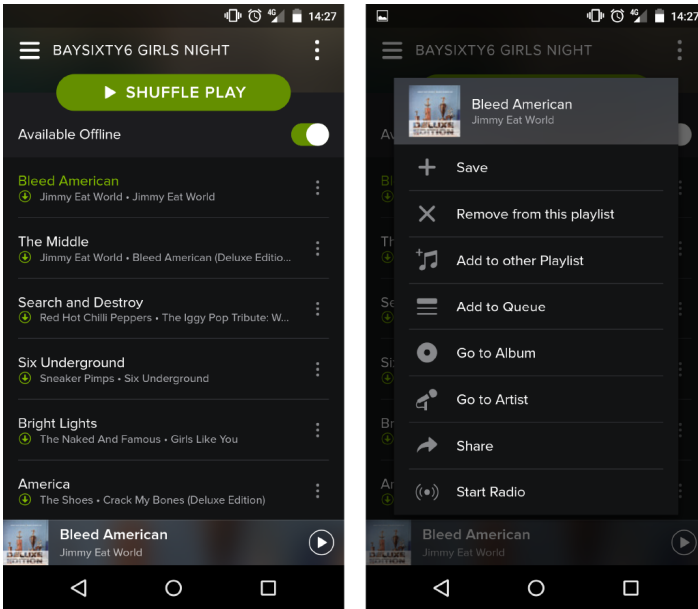
The second priority is for each UI to be consistent to the platform conventions of the device.

Mobile OS UI conventions are well documented in styleguides (e.g., the iOS Human Interface Guidelines<sup>13</sup> and Android Design<sup>14</sup>). There are some key differences—for example, Android users may expect contextual menus on long press, a convention that does not exist on iOS, where the same menu would be displayed on another screen (see Figures 9-21 and 9-22).

13 <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/>

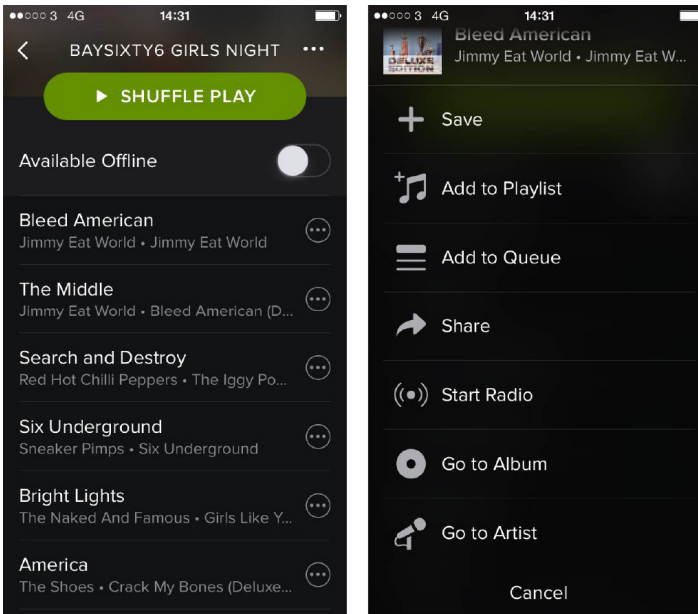
14 <https://developer.android.com/design/index.html>





**FIGURE 9-21**

The Spotify Android app offers a contextual menu for each track



**FIGURE 9-22**

In the Spotify iOS app, the same menu is available on a separate screen

In general, for mobile devices and others that have established platform conventions, following these conventions will make it much easier for users to use your app even if it means some things are done in a different way than they are on any specialized connected devices. A heating control app for iOS must be a good iOS app as well as recognizably part of the heating service. It does not have to be a skeuomorphic representation of a heating controller. But this works both ways: a heating controller does not have to pretend to be an iPhone just because it has an iPhone app. The goal is to ensure that interfaces are appropriate to each device, yet also feel like parts of a coherent service.

Interaction elements such as buttons, menus, and switches should be recognizable according to the platform conventions of the device. An iOS button should generally follow conventions for button styles on iOS rather than trying to look like the physical button on an embedded device.

Most importantly, specific interaction controls should be optimized to the platform. If a physical control uses a wheel, slider, or handle, there is no need to replicate this on a small touchscreen device where it could be harder and more imprecise to use.

For example, the Nest thermostat incorporates a rotating bezel, which is used (among other things) to increase and decrease the temperature. The bezel makes a clicking noise as it is rotated. But the iOS app up/down controls are arrows. The designers could have shown a representation of a bezel onscreen for the user to tap and drag around. But the arrows are a much more efficient and precise control on a touchscreen (see Figure 9-23). Adjustments to domestic temperature are typically within a degree or two, so precision is important to avoid overshooting.

### **Aesthetic styling**

A consistent visual and aesthetic style across all platforms reinforces the perception of a coherent service. Consistent fonts and colors across devices are nice but may not always be practical. For example, the Nest thermostat uses the same font and colors to indicate temperature on the wall thermostat as the iOS app. But this isn't always going to be possible: a cheap monochrome LCD screen won't support a choice of fonts anyway. Replicating the LCD font on a web or smartphone app

may impact readability. It's also a definite retro statement that may not be the look you're after. It's nearly always aesthetically clunky to make a screen design resemble a physical device.

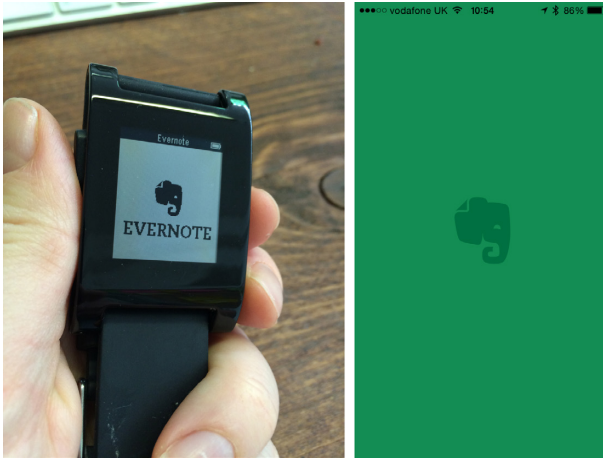


**FIGURE 9-23**

The Nest thermostat and iOS app (showing Celsius temperatures; images: Nest)

Audio is another way to use aesthetic design to create a sense of coherence. Tapping the down/up arrows on the Nest smartphone app produces the same clicking noise per increment as the bezel on the wall thermostat. This is an elegant touch that adds a common aesthetic to each interaction without intruding on usage. It adds to the sense that the devices are a family and helps users form a conceptual model of how the system works.

Where visual elements also convey meaning it is vital that they are used in the same way. This is called *semantic consistency*. To continue with the heating example, you may use red/orange/blue colors to indicate temperature. Or a particular icon might indicate that the water tank is heating up. The icon may be higher resolution on devices with better screens, but it must be recognizably the same thing (see Figure 9-24).



**FIGURE 9-24**

An Evernote icon shown on a smartphone screen and Pebble watch

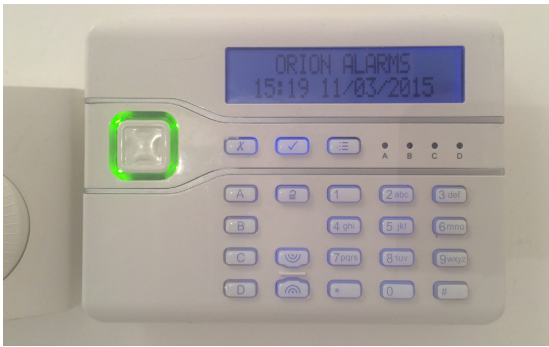
### **Interaction architecture and functionality**

Interaction architecture is the logical hierarchy (or other structure) of the UI as mapped to the controls. This is likely to be less consistent across devices and more platform-dependent. Devices may have different functions in the service. Even where there is an overlap between functions, they may be optimized for different purposes. A wall thermostat might be optimized for small adjustments and switching mode (e.g., turning on the hot water). It might need to support changing the heating schedule, but that's always going to be a better experience on the mobile or web app. In optimizing the thermostat for quick adjustments, the designers might knowingly create a less-good UX for schedule changes. But they might view this as acceptable if users are likely to change schedules on a smartphone or website anyway.

As devices are used for different things, it's not necessarily desirable to group functions in exactly the same way. For example, a mobile or tablet screen can provide one-touch access to many functions, facilitating a broad, shallow functional hierarchy. Fitting the same functions into a heating controller with an LCD screen plus three buttons may require a narrower, deeper hierarchy.

You may also need to use modes, in which the same buttons perform different actions in different states. Modes are typically more difficult to use, but they may be an essential compromise if you're stuck with

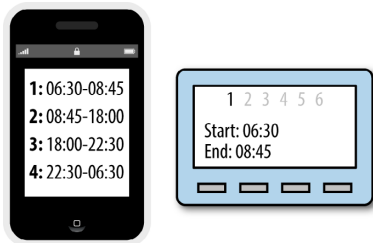
the hardware (see Figure 9-25). Structure your mobile or tablet app to be a great solution for that device, and don't let it be constrained by the limitations of the embedded device.



**FIGURE 9-25**

This alarm panel has a modal UI. In its default state, the numeric keypad is used to set/unset the alarm with codes. When the menu button is pressed, the numeric keys correspond to different items in nested menus (corresponding to functions such as setting zones, adding/removing users, viewing the system log and changing configuration settings). There are no cues on the device itself as to which number corresponds to which function, or which menus are available, so the user has to cycle through buttons to find the option they want. Labeled buttons would have made functions easier to find, but would have added many more buttons and increased the bill of materials.

UIs on different devices don't all have to have the same features, but where they do, the functionality should be consistent. For example, if a heating controller supports a six-phase schedule (six phases throughout the day) but the companion phone app only supports four, users will wonder what happened to the other two and the phone app will run into problems displaying settings that it does not support (see Figure 9-26).



**FIGURE 9-26**

When devices support functions inconsistently, confusion will result

## Consider the most likely combinations of devices

As a designer, you may have to think about design across a large ecosystem of devices. Users may not have all of these. Focus your effort to achieve consistency on the combinations of devices users are most likely to have. To stick with the example of a heating system, all your users might have a controller and smartphone app, but few will regularly use both iOS and Android apps. So it's important that the smartphone apps are both appropriately consistent with the controller. It's less important that knowledge users acquire from using one smartphone app is transferable to the other. For example, the location of the menu button, or the way that system settings are grouped and accessed, need not be the same across mobile platforms, but should conform to the platform conventions (as discussed in "Follow platform conventions"). Few users will use both and those who do are likely to be familiar with both conventions.

## Continuity

In the film industry, continuity editing ensures that different shots flow in a coherent sequence, even if they were filmed in a different order. It would be disrupting to the narrative if a character's hairstyle changed within a scene, furniture moved around, or a broken window was suddenly intact again.<sup>15</sup>

In cross-platform interaction design, continuity refers to the flow of data and interactions in a coherent sequence across devices. The user should feel as if they are interacting with the *service* through the devices, not with a bunch of separate devices.

There are two key components of continuity in cross-device UX. Data and content must be synchronized, and cross-device interactions must be clearly signposted. In our experience, some of the biggest usability challenges in IoT are continuity issues.

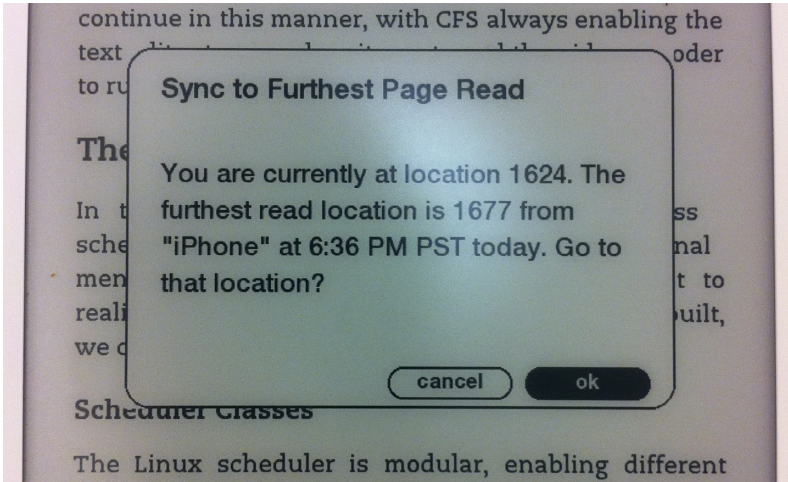
## DATA AND CONTENT SYNCHRONIZATION

It sounds obvious that different device UIs should each give the same information on system state.

---

<sup>15</sup> Spotting continuity errors in movies is a sport. Sharp-eyed viewers share the errors they have spotted on websites such as <http://www.moviemistakes.com>.

Kindle Whispersync is a great example of synchronization. You can switch between reading on different devices—even swapping between the book and the audiobook—and your place in the book is always up to date (see Figure 9-27).



**FIGURE 9-27**

Kindle Whispersync UI dialog (image: Kei Noguchi)

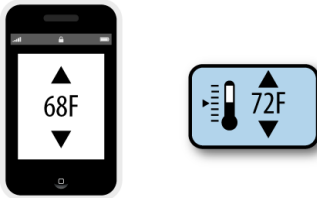
You’d expect this from any other connected device too. For example, say your wall thermostat says it’s 21° C and the heating is on. You’d therefore expect your smartphone heating app to say the same thing, and not to tell you that it’s actually 22° C and the heating is off.

If you turn the heating off from the wall, then you’d expect the smartphone app to reflect that change in state right away, right?

Unfortunately in IoT, as discussed in Chapters 2 and 3, this isn’t always possible. Devices that need to conserve power, such as those that run on batteries, often cannot maintain constant connections to the network as this uses a lot of power. Instead, they will connect intermittently, checking in for new data. This can cause delays and result in situations where some interfaces do not reflect the “correct” state of the system. Network latency is also an issue: it’s possible for the user to know that something has worked before the UI does. For example, they may be physically sitting near a light that they have just turned on and have to wait for a smartphone app UI to tell them what they already know.

To return to the heating example: in the UK, it's common for heating controllers to run off a battery.<sup>16</sup> So a heating controller may need to connect via a low-powered network like ZigBee to a gateway, and only connect intermittently to check in for new instructions. There might be a delay of perhaps two minutes between a setting being changed on the smartphone app and the heating controller receiving that instruction.

This causes discontinuities in the UX. If a user changes the settings on the smartphone app (say, turning the temperature up from 19° C to 21° C), there may be a period of up to two minutes before the heating controller checks in to the service and receives the updated instruction. During this period, the phone UI could show that the system is set to 21° C, and the controller UI will show that it is set to 19° C. If the user is standing in front of the controller with the smartphone app, they will see two conflicting pieces of information about the current status of the system (see Figure 9-28). This violates one of the most fundamental of Nielsen's usability heuristics, visibility of system status: "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time."<sup>17</sup>



**FIGURE 9-28**

In some situations, devices may temporarily report conflicting information about the state of the system

---

16 UK heating engineers prefer battery-powered wireless controllers, as they can be installed easily and sited anywhere without risk that rewiring will be needed. In the UK, mains power is 240V AC and any mains electrical work must be done by a qualified electrician. Even replacing an existing mains controller in the same location requires an electrician. This isn't an issue in the United States, where HVAC controllers typically run on a special low-voltage circuit, making them safe for homeowners to install themselves. This means that in the United States, it's feasible to offer a controller that maintains a constant connection to a WiFi network, and the system can always be in sync.

17 J. Nielsen, "Heuristic Evaluation," in *Usability Inspection Methods*, eds. J. Nielsen and R.L. Mack (Hoboken, NJ: Wiley).



What can you do about this? You could fix this by making the controller check in more frequently. But that would run the battery down within days. Users don't expect to have to change batteries in heating controllers several times a week. So that's not practical.

Your next option is to consider how you can design the smartphone UI to account for this two-minute period. There are two possible approaches.

First, you could show the updated settings the user wanted to apply: the temperature setting of 21° C, even though it might (for a short time) give a misleading impression of the system state. If the instruction cannot be applied for some reason (e.g., temporary Internet outage at the property), you can alert the user and then revert the UI to the old state. In essence, you pretend that it has worked while you wait for confirmation from the controller.

Instagram employs similar “white lies” to make their mobile app feel more responsive. For example, Instagram registers likes and comments in the app UI while the request is still being sent to the service. The user is notified if the action fails. They call this “performing actions optimistically”<sup>18</sup> (see Figure 9-29).

When everything works OK with our heating example, the responsiveness allows the user to feel as if they are interacting with the *service*, not just the phone UI. They have direct control of the heating.

The second approach is to be more transparent about what is technically happening. You show the instruction as being in the process of being sent. This is the approach used by the Lowes Iris system (as also shown in Chapter 3): a status message at the top of the screen is shown to indicate that an instruction is being sent. When confirmation is received that the controller received the instruction, a confirmation message is displayed (see Figure 9-30).

---

<sup>18</sup> M Krieger, “Secrets to Lightning-Fast Mobile Design,” Warm Gun Conference 2011, <https://speakerdeck.com/mikeyk/secrets-to-lightning-fast-mobile-design>.

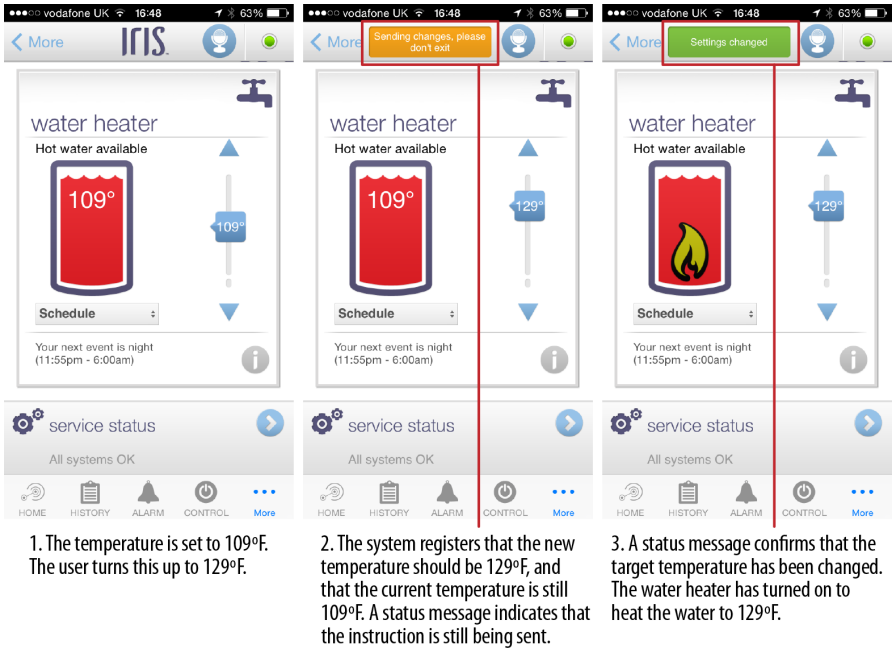


**FIGURE 9-29**

Instagram registers likes or comments in the UI even while the status bar spinner (top right) shows the request is technically still being sent

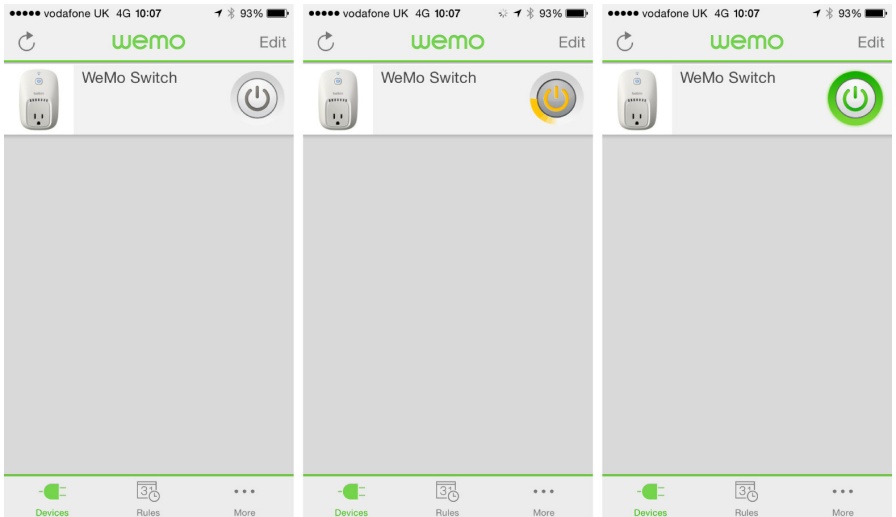
Here, the UI is showing the data as being in the process of being sent. In essence, the system is saying to the user: “thanks for your instruction, let me see whether I can do that.” This requires the user to know a little more about how the system works in order to understand why the instruction isn’t just acted upon. It also introduces the possibility of failure to every interaction, successful or not.

The Belkin WeMo Switch takes a similar but subtler approach (Figure 9-31). When the user turns on the plug, the switch goes from off state (gray) to an intermediate state (yellow with an animation around the switch icon), until it receives confirmation that the action was successful, when it shows as green.



**FIGURE 9-30**

The Lowes Iris water heater UI, showing the status message



**FIGURE 9-31**

The WeMo Switch, showing off, intermediate, and on states

There are no standards here yet, and no right or wrong answer for every situation. In our first example, the primary use case for the system is to support remote access. If the user is out and about and turns the heating on remotely using the phone app, the two-minute delay is not noticeable. The house will be warm when the user gets home. Even if the user is in, heating is the type of system that operates on a timescale of hours, so unlike a light switch, down-to-the-second responsiveness isn't necessarily needed. If the user is standing in front of the heating controller with the smartphone app, it may be confusing, but the compromise may be acceptable.

In other situations, any delay or uncertainty about whether a command has been executed might be dangerous. For example, a person who presses an emergency alarm button must be absolutely confident their call for help has been sent and received. In this case, the UI should not make it appear that the system has received and acted on their command until it has definitely done so.

The frequency with which data is synchronized around the system can heavily shape the user value of the service. For example, dual fuel smart meters monitoring natural gas and electricity usage may report data at different frequencies for each fuel. The device monitoring electricity usage can run on mains power, so it can report data every few seconds. However, it would be dangerous to place a mains-powered electrical device on a gas pipe. So the gas monitoring device will be battery powered. To maintain acceptable battery life, gas data will be reported less frequently than electricity data, perhaps only every 30 minutes.

With live electricity data, users can turn devices on and off and use a display (see Figure 9-32) or smartphone to view almost immediately the energy impact each device had. The system can be used to understand the energy consumption of specific appliances and behaviors, such as boiling water in an electric kettle, or turning on a clothes dryer.

With gas data in 30-minute chunks, it's harder for the user to relate consumption to specific gas-consuming activities in the home. You can't see the immediate impact of turning on a gas cooker. Was the last half hour's consumption high because the oven was on, or because the heating or hot water was in use? So the system data does not directly answer the question "how much gas does my oven consume"? As relatively few activities in the home run on gas as compared to electricity, it's possible for users to make some rough guesses from the data.

For example, if no one is at home but gas is being consumed, that might mean that the heating is on (and thus that the schedule should be changed). But for more detailed insights, the system would need to analyze longer-term patterns in gas consumption data and estimate likely usage by appliance.



**FIGURE 9-32**

An in-home display from Chameleon Technology (image: Chameleon Technology)

When it's not possible for all system data to be perfectly synchronized or “live,” it's important to indicate how old data or status information may be. For example, you might show a timestamp for a sensor reading, or the time that the latest status information was received.

In the energy monitoring example we just looked at, it's important that users understand that the two energy readings are not equally “live.” You could display a timestamp for each reading, but you might also choose to display information in a different format. You might use a line graph for electricity (because you have near continuous readings), but a bar chart for gas, where readings are only intermittent.

It's important to ensure that system status information is as accurate as it *needs* to be for the context of use. In a safety critical system, it should be clear when data may be out of date or an instruction may not yet have been received or acted on. A remote door locking system should not pretend that it has locked the door until proven otherwise! Perhaps

the biggest challenge design-wise is how to design these behaviors for a system that needs to do multiple things with different responsiveness demands, such as heating, lighting, and safety alarms. It's safest to err on the side of communicating what is actually happening, but in some circumstances that may feel inelegant.

## **HANDLING CROSS-DEVICE INTERACTIONS AND TASK MIGRATION**

Cross-device interactions require users to switch between devices in order to achieve a goal. Examples might include syncing data from a wearable fitness tracker to a smartphone, or connecting home sensors to a gateway.

Transitions between devices should be smooth and well-signposted. The word “seamless” is often used in cross-platform UX, but it's probably misleading. Where a task requires the user to interact with more than one device, they *need* to be aware of the seams: the different role of each device, and the point at which the handover happens. This is especially important to help reinforce the user's mental model of the system, and what each part does. The less familiar they are with it (e.g., during setup when devices are new and unfamiliar), the more explanation is required. Next, we set out some key requirements for effective, usable cross-device interactions.

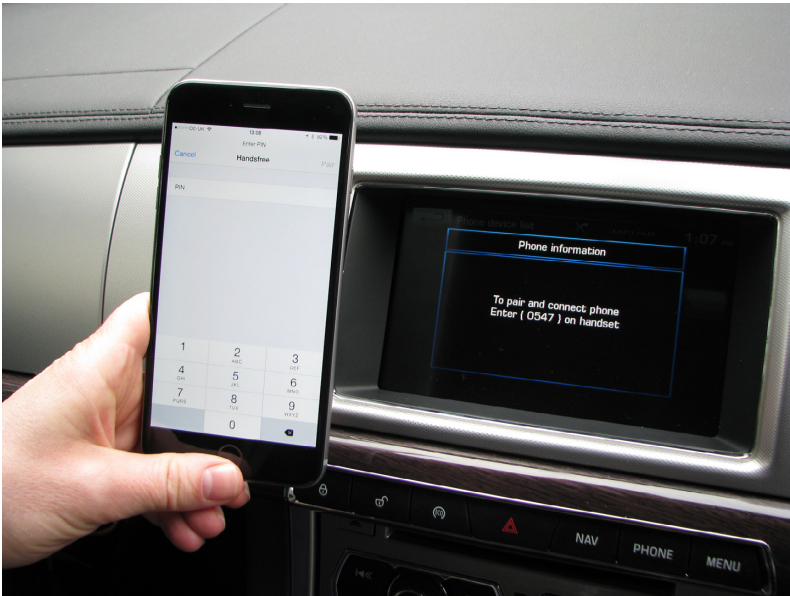
In the first place, the user needs to know that they *need* to switch to another device to complete their intended task. They may have to identify the correct device from among several: for example, there may be several identical light bulbs. Then they need to know what they're being asked to do, and any information that's needed to interact effectively with the other device. They also need to know why they're being asked to switch. For example, are they transferring data, or pairing the devices?

For example, the Misfit Shine syncing process tells the user to place or tap the Shine on the iPhone screen (see Figure 9-33). (Data is transferred over Bluetooth LE but the sync is initiated by the phone recognizing the Shine on the touchscreen.)

The Bluetooth pairing process to connect a Jaguar car to a smartphone displays a four-digit code on the dashboard that needs to be entered in the phone (if not already displayed; see Figure 9-34). (See Chapter 12 for more on Bluetooth pairing interactions.)



**FIGURE 9-33**  
Misfit Shine syncing (image: Misfit Wearables)



**FIGURE 9-34**  
Pairing a Jaguar XF car to a smartphone with Bluetooth

The user also needs to know what reaction to expect from the other device. This is especially important if it has a very limited UI. For example, hitting a button on the Web UI may make an LED flash for two seconds on a motion sensor to help you know which one it is, but you need to know where to look (see Figure 9-35).



**FIGURE 9-35**

Identifying a sensor from the AlertMe web interface (image: AlertMe)

If the interaction is not an integral part of a process (i.e., not something the user *has* to do), provide enough context/content to enable them to decide whether it's important right now. For example, the Pebble Smartwatch can notify the user of new emails, texts, and Twitter alerts, and shows some of the content (see Figure 9-36). The user might not be able to see the whole message, but there's usually enough information to decide whether it's important to get out the phone and read the whole thing there and then. A wearable that only tells you if you have a message, and not who it is from or what it might be, would not offer much over the phone's audio alert or vibrate function.

With a multidevice interaction, it is very easy to lose track of your progress in a task, or for one or more devices to lose connectivity. Where possible, design for interrupted use. Try to avoid locking users into lengthy processes (such as setup) that must be completed in one sitting or in a specific order. Provide some flexibility: if the user has to break off and return later, don't lose their progress—allow them to resume part way through. Guide them back to the parts that need to be completed when they return. For example, a home automation system setup process might require users to associate a gateway with an online account and then pair devices. If the user is interrupted after creating the online account but before pairing the devices, make sure that when



they log in there is a clear route to resume and add devices, not just a blank screen! (See Chapter 12 for more information on designing effective setup experiences.)



**FIGURE 9-36**  
A Pebble notification

### **Broader contexts of interusability**

This chapter has focused on cross-device digital interactions. However, these principles can also be applied to more holistic service design thinking across both online and offline interactions. This broadens the focus of UX to include marketing and sales materials that set expectations of what the product does, packaging and setup guides that shape initial impressions of the UX, as well as customer support.

As with cross-device interaction design, the individual parts can be good, but if they don't work together well the overall experience can still be unsatisfactory or confusing.

You might consider composition when figuring out which setup instructions to put onscreen and which in a print booklet. You would need to consider consistency of language, information graphics, and aesthetics across online and print materials. And you might also need to consider the continuity of any processes that require users to refer between materials. Again, setup is a key example: if your instruction

booklet says “now the LED will blink for 2 seconds,” that’s a pointer to look at the other device (setup is covered in more detail in Chapter 10). Setting user expectations accurately is also a form of continuity: if your marketing materials highlight a feature, it should be easy to find the UI. If not, that’s a form of discontinuity.

The interusability model may not be complete for the broader service context, but we have found it useful for thinking about interactions that span digital and nondigital media.

## Summary

Conventional usability/UX is concerned with interactions between a user and a single UI. Interusability deals with interactions across multiple devices. The aim is to create a coherent UX across the whole system even when devices have very different characteristics.

Users need to form a clear mental model of the overall system, although it can be challenging for them to understand the interconnections between devices.

Designers need to distribute functionality between devices, to suit the capabilities of each and context of use (*composition*).

They also need to determine which elements of the design should be *consistent* across which parts of the system (e.g., terminology, platform conventions, aesthetic styling, and interaction architecture).

Data and content can sometimes be out of sync around the system, causing *continuity* issues. Designers may need to find creative ways of dealing with this in the UI. When interactions begin on one device and switch to another, clear signposting is needed.

# Interoperability

**BY CLAIRE ROWLAND**

Much of what we currently call the “Internet of Things” is not yet much like the rest of the Internet: a network of networks based on open standards. The proliferation of different technical standards means that getting devices to work together is hard. Many devices are locked away in proprietary ecosystems, because frequently that is the easiest way to get them to work. But their lack of interoperability with other devices and systems is seriously limiting its potential value and usability. Users will expect devices to work together, but right now many do not.

Building on the networking concepts discussed in Chapter 3, this chapter explores the technical challenges of interoperability (and some emerging solutions) and their impact on UX.

This chapter introduces:

- Why the current state of the Internet of Things has been dubbed the “CompuServe of Things” (see page 382)
- The concept of interoperability and the problem it poses for IoT (see page 385)
- Ways devices can interoperate (see page 393)
- Emerging solutions for improving interoperability (see page 401)
- The UX of interoperability (see page 405)

This chapter addresses the following issues:

- Why closed ecosystems of devices are not a true “Internet of Things” (see page 382)
- How incompatible networking standards and data formats limit interoperability (see page 388)

- How devices can talk directly to each other, via gateways, or via the Internet (see page 394)
- Why “functioning together” is different from true interoperability (see page 400)
- How web standards may be used to improve interoperability (see page 401)
- The burden on users to figure out which devices can interoperate and get them to coordinate (see page 407)
- The pros and cons of specific versus generic UIs for controlling devices (see page 408)
- The pros and cons of concentrating functionality in single apps, versus having multiple apps (see page 409)

## The CompuServe of Things

Being able to control a device in your home via a mobile app is still quite novel. Most of us have very few connected devices, and perhaps a different app for each type of device. This works OK for now.

But you might, in the next few years, own perhaps 20 connected products. Imagine you have 20 separate apps on your mobile: for the heating, lighting, energy monitoring, security alarm, cameras, baby monitor, smoke alarm, washing machine, slow cooker, and more. Just locating the right app for the action you want there and then becomes a pain.

And the more connected devices you have, the more likely it is that you will want to get them working together. You might want the hallway light to turn on when the alarm goes off to provide illumination for the camera. But right now, many systems don't talk to one another. Each one may be sitting on a separate manufacturer's software platform, with its own frontend app, and often no API to talk to any of the others.

If the systems don't talk to one another, the user has to bridge the gap. Say you want to make more efficient or cost effective use of energy in your house. That requires a combination of energy monitoring data, perhaps other contextual data (like when the house is occupied) and control setting, such as turning off appliances when they are not needed, adjusting heating and hot water schedule, and perhaps running appliances when energy is cheaper. All these data points and controls may

sit in separate systems, creating a lot of work for the user to understand the whole picture, figure out what to do, and make changes to the way things work. (We're not necessarily advocating automation or an autonomous system here, just the ability to offer insights and recommendations alongside user controls.) In the words of the scientist Alasdair Allan, this turns the user into a "mechanical Turk for other people's software."<sup>1</sup>

This isn't much like the Internet as we know it: an open network based on common standards. It's close to the state of online services in the 1980s and early 1990s. Most people who were online then weren't really using the Internet, but a walled garden network provided by an online service provider, such as CompuServe or AOL (see Figure 10-1). (The IoT engineer Adrian McEwen nicknames the current state of IoT the "CompuServe of Things"<sup>2</sup>). These services were closed networks and did not interoperate: for example, if you were an AOL subscriber, you couldn't access content from any other provider, and you could only send email to other AOL subscribers.

Standards for Internet communication developed. By 1994, the World Wide Web was taking off in popularity and public websites were proliferating. One by one, the online service providers became Internet service providers: offering open access to any public website, and the ability to email whoever you liked, whichever their ISP.

Walled gardens briefly reemerged in the early days of the mobile Internet, with mobile operators offering access only to their own portals (see Figure 10-2). But this time it didn't last as long: customers had tasted the open Web and weren't prepared to put up with their operators' bland content portals when the real fun was clearly happening elsewhere.

It's unimaginable today that we would tolerate being restricted to proprietary web content again. Yet the consumer Internet of Things is going through a very similar process. You buy edge devices from a company (often but not always the hardware manufacturer). They provide you with a gateway device (if needed), a mobile/web app, and an Internet service.

---

1 <http://makezine.com/2014/02/03/the-trouble-with-things/>

2 [http://www.mcqn.com/weblog/wuthering\\_bytes\\_slides\\_-\\_risking\\_2\\_things](http://www.mcqn.com/weblog/wuthering_bytes_slides_-_risking_2_things)

**LAST NIGHT, COMPU SERVE TURNED THIS COMPUTER INTO A TRAVEL AGENT FOR JENNIE, A STOCK ANALYST FOR RALPH, AND NOW, IT'S SENDING HERBIE TO ANOTHER GALAXY.**

**NO MATTER WHICH COMPUTER YOU OWN, WE'LL HELP YOU GET THE MOST OUT OF IT.**

If you've got places to go, CompuServe can save you time and money getting there. Just access the Official Airline Guide Electronic Edition—for current flight schedules and fares. Make reservations through our on-line travel service. Even charter a yacht through "Worldwide Exchange."

If your money's in the market, CompuServe offers a wealth of prestigious financial data bases. Access Value Line, or Standard and Poor's. Get the latest information on 40,000 stocks, bonds or commodities. Then, consult experts like IDS or Heinold Commodities. All on line with CompuServe.

Or if like Herbie, intergalactic gamesmanship is your thing, enjoy the best in fantasy, adventure, and space games. Like MegalWars, the ultimate computer conflict.

To get all this and more, you'll need a computer, a modem and CompuServe. CompuServe connects with almost any personal computer terminal, or communicating word processor. To receive an illustrated guide to CompuServe and learn how you can subscribe, contact or call:

**CompuServe**  
 Computer Information Service  
 2100 West Columbia, Ohio 43228  
**800-546-6199**  
 In Ohio, call 614-457-8620

An I&R Block Company

**FIGURE 10-1**  
1980s CompuServe advert (image: AOL)

**vodafone live!**

New For You Messages Tones & Pictures  
 Game Chat News & Weather  
 Sport Erotic Dating Around  
 Entertainment Shopping Love & Life  
 Search My Vodafone

**FIGURE 10-2**  
Vodafone live!, a mobile portal (image: Vodafone<sup>3</sup>)

3 Published at <http://www.w3.org/2001/sw/sweo/public/UseCases/Vodafone/>.

This is how product-based platforms like the Mimo baby pajamas (mentioned in Chapter 7) work. Home automation systems, like SmartThings and Lowe's Iris, may support devices from third-party manufacturers. But development work is required on the platform to build in support for each device. You can't use unsupported devices with the system, nor can you control it with anything other than the app the provider gave you. Things may use the Internet to connect, but they are not internetworked.

In a real Internet of Things, any Thing should be able to connect as a peer to any other Thing (or online service), as computers on the Internet can do. But right now, in technology terms, most Things are vertical stacks of integrated products: they don't *interoperate*.

## What Is Interoperability and Why Is It a Problem?

In information technology, interoperability refers to “the ability to transfer and render useful data and other information across systems, applications and components.”<sup>4</sup> In other words, it's the ability for devices, applications, and services to interact with other devices, applications, and services from different vendors, regardless of hardware architecture or type of software.

The authors John Palfrey and Urs Gasser, in their book *Interop: The Promise and Perils of Highly Interconnected Systems*<sup>5</sup> propose a four-layer model of interoperability.

In their model, the two lowest layers are concerned with the technology (in this chapter, we will focus largely on these).

If two devices can interoperate on the *technology layer*, that means a signal can physically get from device A to device B. If the devices cannot transmit messages to each other, they cannot interoperate in any way.

But the ability to pass messages between device A and device B is not useful unless the two devices can understand each other. This is interoperability on the *data layer*: the devices can speak the same language. (If

---

4 John Palfrey and Urs Gasser, *Interop: The Promise and Perils of Highly Interconnected Systems*, (Jackson, TN: Basic Books, 2012).

5 Palfrey and Gasser, *Interop*.

you've ever had a printer spew out pages of garbage because you used the wrong print driver software, you've seen a data layer interoperability problem.)

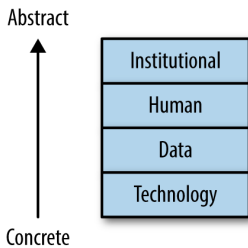
In IoT, the technology and data layers equate roughly to devices being able to speak the same network and application protocols (these are discussed in Chapter 3). In more advanced systems, the use of shared data models (see Chapter 15) may enable devices to work together in smarter ways: this is also the data layer in Palfrey and Gasser's model.

Palfrey and Gasser extend their theory of interoperability beyond technology to consider the people and institutions which that technology supports.

Interoperability in the *human layer* means that the people involved in the exchange of information can understand and act effectively on the information. An IoT example might be the different user needs that have to be taken into account when a product designed for suburban homes in North America is launched in different markets (e.g., differences in language, culture, size of home, and daily activities). (Chapter 5 contains many examples of issues that could be classed as human interoperability challenges.)

At the most abstract, there is the *institutional layer*: the extent to which societal systems can engage effectively around the system. In IoT, an example might be the need to negotiate differing privacy laws in different markets (Chapter 11 focuses on privacy in depth).

This holistic view of interoperability (see Figure 10-3) is a useful reminder that technology isn't just hardware and software, but that it serves human and societal purposes. But as most of this book is about human and societal concerns, this chapter will focus on the technical side: network and data layers.



**FIGURE 10-3**  
Palfrey and Gasser's model of interoperability



In IoT, network and data interoperability means that IoT devices and services can easily discover, communicate, and coordinate with other IoT devices and services, regardless of manufacturer. It also means that devices (or other components) can be substituted: replaced with an equivalent device from a different vendor, as you might replace your web browser or web hosting service with an alternative, without any significant impact on the functioning of the overall system. So whichever manufacturer's connected light switches or bulbs you buy, you can be sure they will work with your other light switches and bulbs, your choice of connected home control application, and even share information on energy consumption with your metering service. Right now, some devices will work with some other manufacturer's devices. But unless they are specifically configured to do so, there is often a sacrifice in terms of functionality (see "How Can Devices Interoperate?" for more details).

Most people involved in the field share a long-term ideal vision of an open, interoperable Internet of Things, but the industry as a whole has not figured out the standards to achieve this. Different standards have often evolved for good reasons: a network protocol that is ideally suited to the home or a smartphone peripheral is not necessarily going to be a good fit for a sensor network. A data model created to describe industrial heating/cooling systems is not necessarily going to be appropriate in the home. (Note that Palfrey and Gasser are careful to differentiate between interoperability and shared standards. Interoperability need not require that everyone adheres to the same standards: that every device uses the same radio or application protocols, that all humans speak the same language, or that all countries adopt the same laws. It only requires that ways are found to translate or mediate between them.)

Security is also a barrier: devices on the open Web are highly exposed, and securing IoT devices is challenging (for reasons set out in Chapter 11).

And then there are business models: a major factor that often mitigates against interoperability.

Interoperability is great for users. It broadens the choice of products, letting them mix and match products and services from multiple vendors to suit their own needs.

But businesses often think that their interests are best served by keeping their technology proprietary and noninteroperable. They are often right—at least in the short term. Companies that perceive they have an initial competitive advantage, such as a differentiated product, or a strong brand or reputation, know that their products will sell (Apple is an obvious example). They may try to turn that into a longer-term advantage by creating products that don't interoperate with those of their competitors, locking customers into buying further products and services from them (of course, they may also argue that their technology is better and that interoperating with inferior systems would degrade the product). Interoperability (e.g., open standards) makes it easier and cheaper for competitors to enter the market. This is good for users and competitors, but bad for the early innovators. If a company is in a position to create a hugely popular set of products and dominate the market, interoperability is not in their interest.

A big company might also choose to collaborate with another big company, so that their products are interoperable with each other's but not with anyone else's. This is effectively forming a cartel to crowd others out of the market.

But in the long term, interoperability helps grow the overall size of the market, which is good for everyone. Many IoT companies are moving toward openness and increased interoperability to create ecosystems of devices and services (e.g., “Works with Nest” and “Works with SmartThings”). These are ultimately more valuable to consumers than the limited product set any one of these companies could create alone.

IoT is on track to become more interoperable over time, but the process of getting there will take time, due to both technical and competing corporate interests.

## **NETWORK INTEROPERABILITY**

In Internet networking and the World Wide Web, interoperability is enabled through universally supported open standards for network protocols (such as the IP stack) and data formats (such as HTML).

Right now, some open standards are used in IoT (such as the HTTP and MQTT application protocols and Bluetooth, ZigBee, or WiFi link networking). But these are not universally supported. Devices and services vary enormously and no single standard is a good fit for every type of system. The diversity of IoT devices and systems is perhaps the

biggest barrier to developing standards. As we saw in Chapter 3, some run on IP networks like WiFi, some on local networks like Bluetooth or ZigBee that don't support IP (at least not yet in commercial products). Some devices need to be constantly connected and exchange lots of data, some need to spend as much time as possible asleep and minimize network activity to maximize battery life. There is no single set of standards that provides the perfect solution to all these needs. Many commentators argue that there never will be.

The lack of standardization exists at different levels of the technology. Proprietary networking, protocols, and APIs or data formats mean that some devices can only work with those of the same manufacturer. Some devices won't even work with different or older versions of the same device from the same manufacturer, if the software or hardware has been upgraded and changed. Where standards theoretically exist (e.g., ZigBee), different implementations of them can mean that some devices can only work with other devices that speak exactly the same version of that standard. Of course, none of them can talk to the many devices that don't speak ZigBee.

Within specific markets, there can be differing networking and data standards too, such as those historically used within commercial HVAC and lighting systems. These were optimized around the needs of the particular market, at a time where there was no conceivable need to share data with other types of system.

And finally, there are limitations at the client device end: the native app that only runs on iOS or Android or Windows. This can be addressed through providing web apps that can run on multiple devices, although these don't always offer the sophisticated UX of a native app.

This lack of interoperability is important to designers because it creates a huge gulf between our idealized expectations of the Internet of Things and the reality of what can currently be achieved.

Users, of course, shouldn't have to care about this stuff. If they buy a connected light fitting, it ought to work just fine with all their other connected light fittings. You don't have to buy all your dumb light switches from the same company because light switches are light switches. Connectivity *shouldn't* change that, but for now, it does.

First, you have to know what networking technology your home gateway supports, such as ZWave or Insteon. Then you have to find light fittings that support that. That's an added layer of complexity that most consumer users could well do without (see Figure 10-4).

SmartThings Benefits Product Blog Community Shop Developers Support

Reviews Works with SmartThings

## Works with SmartThings

Below is a list of products tested by SmartThings and verified as compatible. If you do not see a device you're looking for, it may be compatible with SmartThings but isn't included here because we have not yet tested it. To suggest a device, visit our [Community Site](#).

Sort By

- All Devices
- Home & Family**
- Doors & Locks
- Lights & Switches
- Dangers & Damages
- Motion & Cameras
- Other

Filter By

All Brands

- Aeon Labs
- Belkin
- Honeywell
- Leviton
- Philips
- SmartThings
- Sonos
- Show More

All Protocols

- ZigBee
- Z-Wave
- Local Area Network
- Cloud-to-Cloud

Search for Compatible Devices...

Product	Type	Protocol	Notes
2Gig CT100 Programmable Thermostat	Thermostat	Z-Wave	<i>i</i>
2Gig CT30 Programmable Thermostat	Thermostat	Z-Wave	<i>i</i>
Aeon Labs Key Fob	Button Device	Z-Wave	
Belkin WeMo Motion Sensor	Motion Sensor	Local Area Network	<i>i</i>
ecobee Smart SI Thermostat	Thermostat	Cloud-to-Cloud	<i>i</i>
ecobee Smart Thermostat	Thermostat	Cloud-to-Cloud	<i>i</i>
Honeywell Thermostat (YTH8320ZW1007/U)	Thermostat	Z-Wave	
Jawbone UP24	Fitness Tracker	Cloud-to-Cloud	<i>i</i>
Logitech Harmony Home Hub	Hub	Cloud-to-Cloud	<i>i</i>
Netatmo Weather Station	Thermostat	Cloud-to-Cloud	<i>i</i>
RCS TZ45 Thermostat	Thermostat	Z-Wave	
SmartThings Hub	Hub	ZigBee Z-Wave Local Area Network	<i>i</i>
SmartThings SmartSense Motion Sensor	Motion	ZigBee	
SmartThings SmartSense Presence Sensor	Presence	ZigBee	
SmartThings SmartSense Temp/Humidity Sensor	Temp/Humidity	ZigBee	

**FIGURE 10-4**

The SmartThings website lists compatible devices (shown here: the home and family category; image: SmartThings)

The simplest UX designs right now come from single platform experiences, such as the Philips Hue bulbs or Withings health and fitness ecosystem. But this limits users to devices from a single manufacturer if they want things to work together. Do you want to have to buy everything in your house from Apple, Samsung, or Philips? It also makes the cost of switching to a different platform very high. Old devices won't work with the ones from the new manufacturer and must be replaced. Data is at stake, too: a user who replaces their activity monitor or bathroom scales with a device of a different brand may lose their data because it will almost certainly be tied to the original vendor's platform.

All of this is hampering the potential of the IoT through limiting the exchange of data and functionality. People and devices should be able to communicate and work together regardless of manufacturer, model, or network technology.

Part of the problem is that there is currently a financial disincentive for vendors to create interoperable products. The business models of big players such as Apple, Google, and Microsoft depend on proprietary operating systems and/or retaining users and their data. There is no incentive to enable devices to work with others outside the walled garden: as it stands, this would undermine the business model.

There is also a design disincentive, especially for a company that considers great UX a differentiator. Interoperating with third-party devices often results in a clunkier user experience and often a reduced feature set. This can happen because different devices use different formats for data, APIs, or network device profiles (as used in ZigBee). Third parties may not be able to access all the data or functionality on the system.

As an example, for a while, the home automation platform Insteon unofficially supported the Nest thermostat, but offered only basic controls. Users who wanted to control their Nests alongside other connected home devices either had to sacrifice using some of the more advanced functionality, or switch to the Nest app for that (and then risk conflicts when two apps are trying to control the device). Insteon now works fully with Nest, but this type of integration can be effortful to do.

As the original equipment manufacturer (OEM), you may also want to discourage the user from using third-party apps that you think are not as well designed as your own or may cause unintended problems.

Devices may be linked at the Internet-service level via APIs. If this is the case, all interactions must go via the Internet where responsiveness could be an issue. For more details on different system architectures for interoperability, see “How Can Devices Interoperate?”

It’s widely acknowledged that the current state of affairs is a mess. Some organizations are trying to solve it by creating platforms that consolidate various network standards and devices. For example, the Wink gateway (see Figure 10-5) and app platform supports multiple networking standards (e.g., WiFi, ZigBee, Z-Wave, Bluetooth, and Lutron Clear Connect).



**FIGURE 10-5**

The Wink hub (image: Wink)

It can be time consuming to create systems that allow different devices to work together. For a connected home platform, every single third-party device on the platform must be individually configured to work optimally with the system. It is not enough to have the correct radio type on the hub; the device must also be associated with a software profile, which specifies the feature set it supports. There are standard profiles, but they do not necessarily support the range of functionality that modern devices offer. For example, ZigBee offers separate contact and temperature sensor standard profiles. A device cannot be associated with more than one profile. So even a simple device that can sense both open/closed status and temperature will need a proprietary device profile. Similarly, the ZWave profile for a thermostat is based on old-fashioned devices that only support manual temperature control for heating. Devices with timers or air conditioning require a proprietary profile.

The more proprietary profiles there are, the less interoperable the system. But if you only support standard device profiles, you may well not be able to support the full range of functionality users expect from a device. All ZWave devices are mandated to support basic on/off functionality, so any ZWave device connected to a gateway will in theory do something. But this may not be very useful: a ceiling fan, for example, might only be able to be switched on to 100% power or off. A ZigBee device will do nothing unless it has a suitable profile.

Testing and reliability also becomes a headache as there are so many potential combinations of devices that the user could have. Any combination could cause unexpected errors, but it is unfeasible to test them all.

## INTEROPERABLE DATA

Getting devices to connect is only part of the problem. Are they able to understand each other, or are they talking gibberish? Local networking protocols designed for specific purposes (like home automation) may provide a basic vocabulary of commands: so, for example, any Z-Wave device can understand an “on” or “off” instruction.

But there are no standards for more complex data. Fitness trackers may measure activity in a number of ways: perhaps minutes of activity or steps or distance traveled over time (for running). Nike+ has a proprietary measure of activity called “FuelPoints” (see Figure 10-6). Or lighting brightness may be measured in lumens or candela across different sensors and systems. So, combining and comparing data about the same things when they are measured or organized in different ways is yet another interoperability challenge. This is the realm of metadata. It is possible to program devices to translate where the units have an equivalence, for example, where the system knows that 5,000 steps is the equivalent of 2,000 Nike FuelPoints. But if one system is measuring heart rate and the other is counting steps, the relationship between the two is much more complicated. Metadata models are discussed in more detail in Chapter 15.



**FIGURE 10-6**

Nike+ measures activity in FuelPoints (image: Nike)

## How Can Devices Interoperate?

In this section, we’ll look at the different technical levels on which devices can work together, which has an impact on UX. We’ll also explore the distinction between enabling devices to work together in a closed system and making them truly interoperable.

## INTEROPERABILITY AT DIFFERENT LEVELS OF SYSTEM ARCHITECTURE

Different approaches to interoperability can have different implications for the user experience. As we saw in “How System Architecture Affects UX” in Chapter 3, devices can connect and share data on different levels. Engineers have sought to create interoperability options at all of these levels. For the same reasons that system architecture can have implications for UX, so can the level on which devices interoperate. We set out some examples of these in the following sections.

### One edge device connects directly to another

Two devices supporting a common networking technology (e.g., WiFi, Bluetooth) can connect directly to each other over a local network link. In a simple case, one device may only need to register the presence of another in order to trigger an action. In the Ninja Sphere home lighting demo from Ninja Blocks (see Figure 10-7), the lighting detects the presence of a Fitbit’s Bluetooth LE profile and turns lights on and off as the user moves around the house.



**FIGURE 10-7**

“Follow Me” lights demo (images: Ninja Blocks)<sup>6</sup>

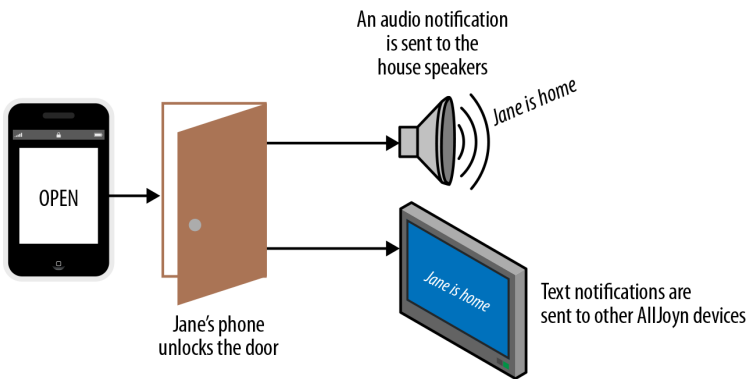
In a more complex case, actual data or instructions will be exchanged. AllJoyn<sup>7</sup> is an open source project initiated by Qualcomm. It provides a software framework, networking standard, and set of core system services to enable edge devices of different manufacturers to talk to each other and other software applications, as long as they all support

<sup>6</sup> <https://www.youtube.com/watch?v=X6o4TGDgTVA>

<sup>7</sup> <https://allseenalliance.org>



AllJoyn. AllJoyn services allow devices to announce their own capabilities on a network, and thus discover and connect to other nearby devices with the capabilities needed to fulfill a particular task, such as using screened devices to act as control panels for nearby headless (nonscreened) devices, or routing audio (such as a doorbell alert) to the most appropriate speaker (see Figure 10-8). For example, an August door lock can broadcast a text notification around the house that the door has been opened so other devices can greet you. It can also translate this to audio and send it to Panasonic speakers. Even if the manufacturers of two devices have never worked together, the shared AllJoyn framework allows for unexpected interactions between devices.



**FIGURE 10-8**

Image illustrating AllJoyn scenario

If you can trust the devices to pair reliably, device-to-device connections should be quick and responsive. In an example such as AllJoyn, there is the potential to streamline how user inputs and outputs are handled around the home. This could reduce the number of devices in the home that require specific capabilities: for example, instead of having to build speakers into every device that needs to output audio, you can have one good set of speakers in every room.

### **One edge device connects to another via local gateway/hub**

In this scenario, devices that use different networking technologies connect to a gateway device or hub that contains multiple radios. This allows instructions to be passed between the two. This setup could be used to control devices of the same type running on different network types together via a single app. For example, you could turn on both

Insteon and ZWave light switches with a single master switch or lighting rule. The Wink hub (gateway), described earlier, is an example of this, as are SmartThings, Staples Connect (see Figure 10-9), and Lowes Iris. One of the most ambitious was Revolv, whose hub had seven radios.<sup>8</sup>



**FIGURE 10-9**  
The Staples Connect hub

If there's some intelligence in the hub, more complex instructions can be passed from one device to another. If a ZWave door lock is opened, instructions could be sent to ZigBee light bulbs to flash in red or green. In this example, some system intelligence and configuration is required in the hub to know that the door lock opening should trigger the bulbs to flash, as the two edge devices aren't smart enough to handle this themselves. But the fact that the two edge devices run on different network types isn't a barrier to using them as part of the same system. This is the typical approach in the home automation world: allowing users to access and control heating, lighting, security, cameras, and many more types of home system from a single platform, and a single web/mobile app.

A consumer setup like this would be intended to be used mostly with an Internet connection. So the intelligence that tells devices *how* to work together might be partly or wholly in the cloud (so would not work if the connection was lost). But if there is appropriate intelligence in the

---

<sup>8</sup> Revolv was acquired by Nest in 2014. Their hubs are still supported but are no longer on sale.

gateway, then instructions can be passed between the two devices without relying on the Internet. So, for example, the automated rule that turns on some lights after dark if no one is home could still run even if the Internet goes down.

Device-hub-device interactions should be quick and responsive, as they run over local networks. But sometimes joining two devices that support different networking technologies can lead to reduced functionality.

### **One gateway connects to another**

It's possible for one gateway to connect to another. This allows two systems to be integrated over local networking, for a responsive experience, but also for the devices to continue to connect to their own manufacturer cloud. The SmartThings gateway can connect directly to a Philips Hue bridge. Any connected Hue bulbs will appear directly in the SmartThings app and continue to be available to the dedicated Hue app.

### **Edge devices connected via a cloud platform**

In this scenario, data and controls for edge devices are integrated via an Internet service. For example, a fitness tracker (connected via Bluetooth to a mobile device) and a WiFi bathroom scale (like Withings) would both connect to the Internet via different means but share data with the same Internet service (so the user would access the data from both via the same web or mobile app). The same fitness tracker could even display a friend's step count (from their fitness tracker) that day as a spur to competition (if you both agreed to share data). Both devices connect to different mobile devices, the integration happens via the manufacturer's cloud.

Interactions with devices connected via a cloud platform will inevitably be subject to the inherent latency of the Internet. Data can't be shared between devices if either loses Internet connectivity. So this method best suits devices with a reliable connection, or services (like weight tracking) where data doesn't necessarily have to be live, just updated regularly.

### **Service to service over Internet**

In this scenario, the integration is between Internet services. A service provider makes an API available to allow other services to retrieve data and perhaps send commands. These APIs might be totally open—for

example, the Xively public API for open sensor data. Or they might be accessible only to others participating in a developer program, as is the case with Nest and Philips Hue. Connected Mercedes-Benz cars can now notify a Nest heating controller that the driver is on the way home, triggering the heating to come on automatically (see Figure 10-10). The notification goes via the Mercedes-Benz Internet service to the Nest Internet service. Or the Nest Internet service can notify the LIFX lighting service that the Protect has detected smoke, triggering the bulbs to flash red to indicate danger.



**FIGURE 10-10**

The Mercedes-Benz dashboard app showing Nest integration (image: Mercedes-Benz)

Cloud integration works well in the cases where Internet-grade responsiveness is acceptable. Interactions with interconnected services will not only be subject to Internet latency, but have more potential points of failure (devices, gateways, Internet services). APIs may impose loads on how frequently other services can poll (check in) for changes, which might also create potential delays. For example, it can take up to 15 minutes for IFTTT to check in with Gmail. If you were hoping to notify yourself of emails from someone important by making a Hue light bulb change color, you might find it's not the fastest way to find out. If you need things that are near each other to connect immediately, a local network is more reliable.

Nest seeks to mitigate this by promising “real-time” communication between their APIs and their devices, so in theory, service-to-service interactions can be quick. However, in many cases, particularly where an API is public, the provider will have imposed limits on how often it can be polled so this will not always be the case.

As noted, services may capture, store, and handle data in quite different structures (e.g., a RESTful API for pollution sensors might be structured as country/area/sensortype or sensortype/country/area, and fitness trackers may measure activity in minutes, steps, or heart rate), leading to APIs that, despite doing similar things, may be structured in quite different ways. This means that any developer who wants their system to access a third-party API will need to spend time understanding what data and functions it offers and how it is structured, and program their own system to request data and pass commands in the right way. Right now, the inconsistency of data structures means that it's rarely easy for a service automatically to discover other services and request data.

### **Products and services can interoperate on several levels**

There are currently lots of companies seeking to become the de facto gateway/cloud integration platform for the home, through a combination of gateways that support multiple networking technologies and cloud services that talk to other cloud services. It's quite possible for a system to combine both approaches.

Manufacturers may wish their devices to continue to connect to their proprietary cloud first, even if those devices could technically connect to a third-party gateway. This allows them to maintain their relationship with the user, retaining their data and providing their own services.

Sometimes integration providers (such as SmartThings) will take a fairly open approach to third-party devices, making developer kits and APIs available to third parties to integrate their own devices onto the system. Approval may be required before the device is officially declared "supported," but in principle the process is fairly open. Other companies (such as Staples Connect or Lowes Iris) may only integrate products made by manufacturers with whom they have a business partnership.

This latter approach may mean that the experience of adding and using a supported device will be more consistent with the general platform UX. But it means that other devices won't work (or won't work well) even if they use a protocol that the hub supports.

## FUNCTIONING TOGETHER VERSUS TRUE INTEROPERABILITY

Platforms that connect devices from different manufacturers together or support different radio protocols are an improvement on wholly closed manufacturer ecosystems. But they aren't anywhere nearly as interoperable as, say, the Web. You can't plug any light switch into any connected home system and expect it to work. But you can point any browser at any HTML-based website and (barring inconsistent support for some advanced tags) it should do something with it.

There are advantages to closed systems: for one, they are easier to secure. Security for connected devices poses very complex challenges and solutions are immature (see Chapter 11 for a user-centric introduction to security). A closed system allows you to control who and what can connect to the system, and what they can do. This also improves reliability. Devices can conflict in unexpected ways, causing odd behaviors. Standards (like device profiles) that are ostensibly the same may have been implemented slightly differently. If you control the devices that can work with the system, you can configure them to work together sensibly, and QA test them together. (However, as you add support for more devices, the potential combinations that need to be tested will rapidly increase and it will at some point become unmanageable to test everything.) Finally, a closed system gives you more control of the UX. If the user adds a heating controller you support, you can offer them the perfect UI for that specific controller. A generic heating controller UI, designed by a third party to support a range of devices, might be a poor fit for your specific device. For example, the UI might use different terminology such as “schedule” instead of “timer” or have only a subset of the functionality that the controller supports.

Some platforms are semi-closed ecosystems, where access is granted to selected third parties to build services that integrate with the platform. The Nest developer program mentioned earlier is an example of this, sharing data over APIs with other manufacturer clouds, such as Jawbone and Mercedes-Benz. This provides some interoperability—more devices and services are supported. But both parties retain enough control to ensure security, reliability, and UX aren't compromised.

Some platforms are entirely open, like Xively: anyone can access any data on the Xively public API without requiring permission. Devices connected to Xively are generally at lower risk: the most you can do with many of them is read data that the sensor owners have chosen to

share publicly (see Figure 10-11). Xively is interoperable in the sense that the data is publicly available and well structured, so it's easy for third-party developers to work with. You can program your system to retrieve data from Xively. But that code will be specific to Xively. If you want to retrieve temperature data from sensors on Xively and also sensors on another service, you'll probably need slightly different code for each service, because the structure of the data might be different.



**FIGURE 10-11**  
The Air Quality Egg map of sensors (image: Air Quality Egg)

## How Can We Improve Interoperability?

Standards are like toothbrushes. Everybody agrees you should have one, but no one wants to use yours.<sup>9</sup>

Interoperability can be improved through addressing inconsistencies in networking (ensuring devices can connect) and data (ensuring they can communicate and coordinate). Inconsistencies could be ironed out

<sup>9</sup> Original source unknown, variously attributed to Joe Croser and Anita Golderba. Via Rick Bullotta of Thingworx.

by everyone conforming to a common standard, such as using ZigBee networking in the home or using a common semantic structure for data, but this probably isn't realistic given the wide range of applications covered by IoT. A network of several thousand solar-powered air quality sensors will run on low-bandwidth connections and have a high tolerance for latency and reliability issues and missing data points. A medical monitoring system for a critically ill patient and a home lighting system will have very different needs.

A more sensible approach is to accept that different standards will continue to exist and work around that.

At the network level (of getting the things connected), gateways may contain more radio chips so that more edge devices can connect to them. Earlier, we mentioned the Revolv home automation gateway that has seven radio chips. But additional radios (like Bluetooth and ZigBee) could also be built into devices most of us already have in the home, such as broadband and WiFi routers.

More powerfully, methods are emerging for devices and services to communicate that are independent of the type of network they use. The 6LOWPAN project defined mechanisms for IPv6 to be transported over low-bandwidth wireless networks, meaning that IP networking could be extended to low-powered edge devices. This makes them capable of being directly addressable on the Internet and enables the use of application protocols such as HTTPS or MQTT. So your low-powered light switches can speak the same "language" as your WiFi-enabled speakers, making it much easier to coordinate the devices around you. 6LOWPAN support is beginning to appear in commercial consumer products, such as the Tado heating controller:

Fitting a building for "IoT" shouldn't mean fitting a hundred different custom gateways. It should be more like WiFi deployment.

**TOBY JAFFEY, CTO OF 1248<sup>10</sup>**

Within a service or platform, you might still want to treat devices of the same type that were made by different manufacturers or run on different network types as equivalents. Device abstraction is an engineering approach that maps actual devices to virtual representations of devices.

---

<sup>10</sup> Personal communication, via Twitter (@tobyjaffey).

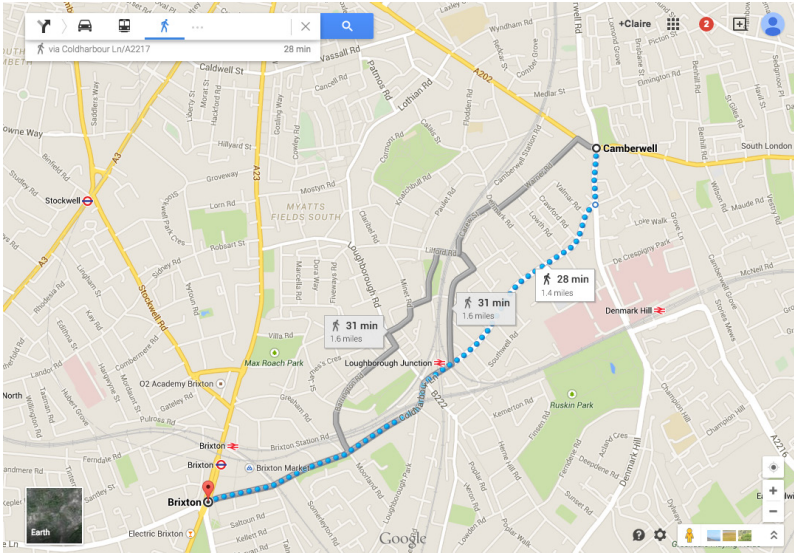


Your door and window sensors may be made by different manufacturers, but the system can recognize and treat them as instances of the same thing. When you add a new door or window sensor, the system recognizes it as a door/window sensor and groups its data with your other door/window sensors no matter who made it or whatever network it runs on.

To be truly interoperable, services also need to be able to discover useful data and functionality within other services. That's not easy when APIs, security, and application data formats are inconsistent. Imagine you were staying in an Airbnb apartment with connected devices such as the door lock, heating controller, entertainment devices, and lights. To control these using current technology, you'd have to download one or more dedicated smartphone apps provided by the manufacturer(s) of the devices in the home and then log into the homeowner's accounts (or special guest accounts they set up) in order to use them. You might think this was too much hassle. Wouldn't it be better if your own home control app could automatically discover the devices around you and offer you the home controls you needed for the duration of your stay? Or you could just discover the devices around you using a web app?

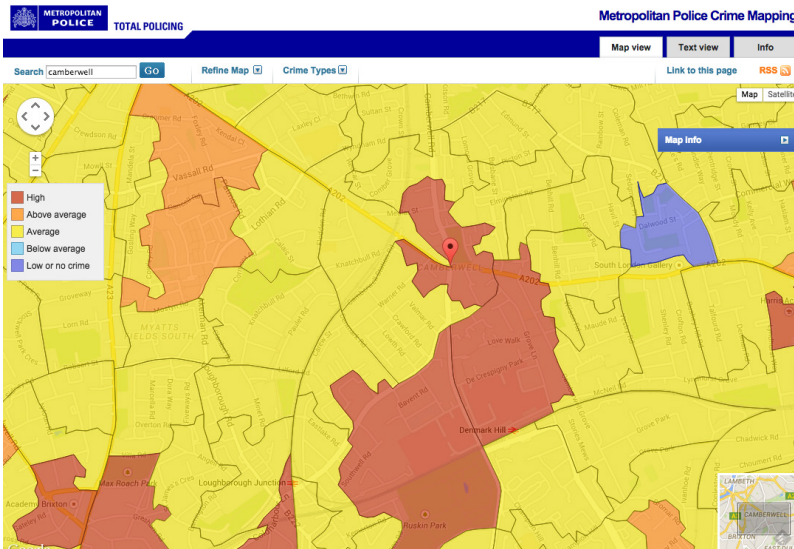
Or you're walking back to the apartment late at night in an unfamiliar city and want to find a well-lit route where there are likely to be other people. Traditional mapping apps may tell you the route, but don't provide any contextual information to help you decide how safe you may be (see Figure 10-12).

Local live data on city street lighting would be useful. It would also be useful to have data on current road traffic density—are there people driving around or is this street deserted? Historic crime levels might be useful, too (see Figure 10-13). You may want to know whether it's safer to try to get a bus, but you have no idea of the routes or when the next one might be due. This is data that may be available publicly but isn't easily accessible. Each city has its own, separate, public transport apps, but you have to know where to find them and then download them. The data on street lighting, traffic, and crime could be online somewhere but probably in separate places, and anyhow, it won't be easy to get the information you need, which is “which roads on my way home are well lit, not deserted, and not crime hotspots?” What you need is one service that you can rely on to retrieve local data, wherever you are, and crunch it all to give you the answer you need. And what that service needs is a way of finding all this information.



**FIGURE 10-12**

Google Maps doesn't yet provide information that would help you understand a route's safety (image: maps.google.com)



**FIGURE 10-13**

The London Metropolitan Police maps show historical crime rates by local area (image: maps.met.police.uk)

The HyperCat standard (based on HTTPS, REST, and JSON) seeks to enable this kind of discovery. It provides a very simple, light API catalog layer that allows apps to search on a hub or IoT service to discover what resources (devices, data) are available that the app can use, or to look for very specific data types (even if they don't understand any of the other data). So an app that is simply looking for lighting brightness or traffic density data can find that, even if it's buried in a load of other irrelevant information about lighting maintenance schedules or traffic light cycles that the app can't understand.<sup>11</sup> HyperCat does not aim to solve all the problems of interoperability at every level (such as common semantics or data formats), but just to address the basic problem of enabling apps/services to discover useful resources on other services.

Ultimately, a complex system will need ways of discovering and dealing with very different, and even unfamiliar devices—devices that may be connected to it directly, and things it discovers out there on the network. Sensors may have different sampling rates or capture different measures, and actuators (even those that do similar things) may have different capabilities. Knowing what something can do, and how it captures/structures data is one challenge. Coping with changing network resources is another. Things can go online or offline or cease to exist. New devices, that the system doesn't know anything about, may appear. In Chapter 15, we cover the use of data models: a loose group of methods seeking to enable interconnected systems to understand each other and coordinate.

## The UX of Interoperability

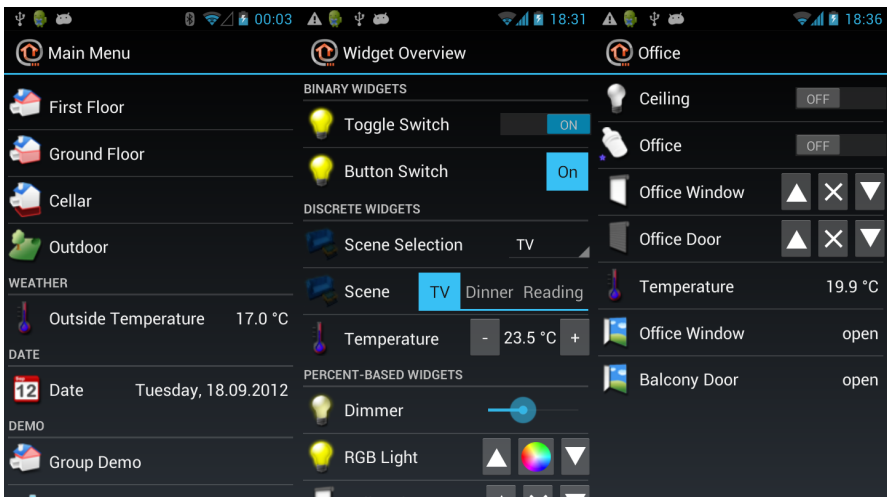
Currently, consumers are buying devices to fulfill specific needs: lighting (e.g., Hue or LIFX), music (e.g., Sonos), heating or cooling (e.g., Nest or Tado). People may have trouble understanding networks (as we saw in Chapter 9), but in our experience it does seem obvious to them that devices should work together. If the heating system knows I'm on vacation, why wouldn't the lights? They will also expect devices to work with equivalent devices from other manufacturers. You may have

---

<sup>11</sup> <http://1248.io/casestudies/hypercat.pdf>

a few full-featured Hue or LIFX light bulbs, but you also want these to work with cheaper, more basic bulbs from other manufacturers, or connected light switches or smart plugs on lamps.

Interoperability is really about reducing the amount of human intervention involved in getting things (or services) to work together: to return to Alasdair Allan’s quote, to avoid both programmers and users becoming “mechanical Turk for other people’s software.” Right now, it’s often not possible to get devices working together unless they have been programmed to do so by the manufacturer. Where it is possible, it’s often difficult to do so and the UX and functionality of the devices may be compromised. Open source alternatives may exist, such as Open HAB open source home automation software (see Figure 10-14), and fulfill a need for those who do not want to entrust their homes or lives to a corporation. But they require a degree of technical skill that most consumers will not possess.



**FIGURE 10-14**  
Open HAB home automation software (image: Open HAB)

## UX ISSUES IN INTEROPERABILITY

In this section, we outline the key challenges that interoperability poses for both users and designers. We’ll also look at why, despite those challenges, interoperability is ultimately in the user’s interest.

## Knowing which devices will talk to each other

Just knowing which things could potentially work together can be hard. For an iPhone or Android phone user buying a wearable, it's fairly simple to find out whether the device supports your phone, as there's likely to be an easily recognizable Apple/Android icon on the box to tell you whether the app will work on your phone. You might need to know whether your phone supports Bluetooth LE, but most modern smartphones will.

If you want to add a device to a connected home system, it's a bit more complex: unless a device has been explicitly sold to work with your system, users have to know what type of radio a device uses (e.g., Insteon, ZigBee, ZWave, Lutron Clear Connect). This is more of a pain: the icons for these systems are less easily recognized (see Figure 10-15).



**FIGURE 10-15**  
Icons for different radio types

With 6LOWPAN bringing IP networking to edge devices, more and more will be able to talk, but this is a long way off becoming standard. And you'll still need to make sure there's a suitable radio for your device to connect to.

## Getting devices to coordinate

Even if you can add two devices to the same gateway or Internet service, that doesn't necessarily mean that they will work together in clever ways. The example given previously of the Nest Protect causing LIFX bulbs to flash red when the alarm is sounding has been configured in the service. If the user has the devices set up, they will most likely only need to enable this option in the app. But on another connected home platform that supports a wider range of hardware you might have to configure your own rules in order to achieve a similar effect. This is a much more complex activity, more like programming.

To make it simple for users to configure interoperable devices, system designers either need to build in lots of preconfigured options, and/or build more intelligence into the system so that it “knows” what devices do, and what kinds of coordination between them may make sense.

See Chapter 15 for more information on designing programming-like experiences, options for making the system smarter, and the pros and cons of different approaches.

### **Generic versus specific UIs**

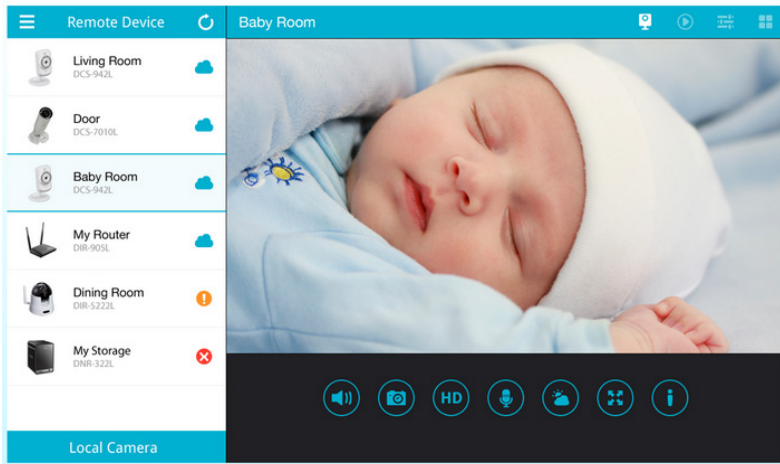
A specific app for your device allows the app UI to be optimized to exactly the functionality your device offers. You can offer a rich experience, specialized to the needs of your devices. An integration platform allows many devices to be accessible via the same app, with a generic UI for lots of devices, regardless of manufacturer. This may not provide access to sophisticated extras that are unique to your device (such as firmware upgrades).

For example (at the time of writing), the Staples Connect app allows users to view video from D-Link cameras, but not to zoom, adjust brightness, take a snapshot, or view the image in landscape mode,<sup>12</sup> all possible in the dedicated D-Link app (see Figure 10-16). The Hue lighting app offers a party mode, but the generic UI in the Revolv app (which supports Hue) does not. This example shows how you may have to consider how to show status information in the generic UI if it does not support functionality the user activated in the proprietary app.

If you’re designing a generic UI, you need to be flexible enough to cope with unexpected devices and handle them as much as possible. For example, all Z-Wave devices are mandated to work somewhat even with controllers that have no idea how to handle them. Most of the time that is just setting a single state that will hopefully do something useful. But while this is helpful for an advanced user, it will confuse everyone else.

---

<sup>12</sup> As reviewed by <http://bit.ly/1EzAAzH>.



**FIGURE 10-16**

D-Link's dedicated camera app works in landscape mode and allows users to take snapshots (image: mydlink.com)

### **Organizing devices and accessing controls**

If you have a lot of connected devices, you may, as things stand, end up with lots of apps. It can be hard to find the app you want. And as devices will be tied to the manufacturer's platform, it's likely that they won't necessarily interoperate or will do so in only limited ways. If you have lighting from several manufacturers, you'll have several apps.

Having one app per manufacturer effectively means that your devices are organized by manufacturer. But mostly, users don't care who made a device as much as they care about what it does. Grouping devices by manufacturer isn't a great fit for their needs. In some situations, users may want to see all the devices that do the same thing—for example, all the devices that measure temperature or all the devices that light the home. Or they might want to see all the things that support the same activity, such as leaving home in the morning. And often they may want to see things that work as systems, such as the heating controller, boiler, and hot water tank. It's very rare that they will want to see them organized by network protocol: all their ZWave devices in one place, and all their ZigBee devices in another. But these are the kinds of groupings that poor interoperability unintentionally creates. (Sometimes the right UI will not be device centric at all. Chapter 15 discusses ways of organizing large sets of devices in more detail.)

The idea of a single app to control many devices, especially around the home, is superficially appealing. This approach can work quite well with a moderately sized set of devices.

But the UI may be less rich, as already discussed. And when you have a lot of devices, a single app could also become an unwieldy way to access them. The user could easily end up digging around several levels deep to find the control they need in a mobile app. Four taps to turn off the hall light is rather a lot. Of course, you can provide quick access to key functionality on a homescreen, but there will be a point where this approach won't scale to all the controls the user might need frequently.

So there may be an advantage in different frontend apps: more key functionality can be made quickly accessible. In an ideal world, apps need not be manufacturer groups: you might have one for lighting, one for security, and one for HVAC and hot water. But if you want to coordinate functionality across multiple apps, where does that go? If you want the security alarm to turn on certain lights if the alarm goes off, does that sit in the security app or the lighting app? This is an issue right now, in our manufacturer-centric app world. You may go into your Hue app to authorize several other services to access this, your SmartThings app to authorize some other services, including Hue... and so on. There is no centralized way to manage authorizations. This can lead to problems keeping track of which service can do what, and might lead to problems, where one service is receiving conflicting instructions from two different services.

One option could be to offer separate control apps for lights, heating, security, and so on, then a “settings” app that controls how they all work together and handles overall preferences. But the user would then have to install and maintain several apps—which can be confusing at point of setup. And doing this across manufacturers would depend on a high level of integration.

Finally, consider whether you need to set the user's expectations about how things may work. In an ideal world, users should not have to care whether devices in the same place may be on the same local network or connected via the Internet. But you might choose to differentiate them in the UI if there are likely to be latencies in the experience, to set expectations and provide an explanation of any delays reasonably.



## Interoperability puts users in control

Closed ecosystems of closely integrated hardware and software may tend to offer better UXes for now. But many people feel strongly that closed ecosystems lock users off from doing whatever they like with their own data and devices, and prevent developers offering them more choice. Fierce proponents of Android often take this view of iOS in the mobile sphere. Many consumers don't yet feel strongly about this when it comes to their mobile phones, but this may change as more and more of the everyday things around them become connected. True interoperability means that we can choose to be in control of our things without being beholden to an authority that limits what we can do with them. It means you don't have to buy all your devices from the same manufacturer, or throw away perfectly working devices just because they are incompatible with your new home. It means you're not dependent on your car's manufacturer to keep it running, so they can't stop you using it if you miss a subscription payment. And it means that your heating controller need not stop working because the company that made it went out of business.

## Summary

The Internet as we know it is based on open standards, but many IoT devices are currently locked away in closed ecosystems.

Interoperability is the ability of devices, applications, and services to interact with other devices, applications, and services from different vendors, regardless of hardware architecture or type of software. As things stand, the diversity of networking standards and data formats used in IoT systems means that many systems have limited interoperability. This limits the potential value of IoT and inhibits the quality of the UX.

Devices can interoperate in different ways. Edge devices can connect together directly, via a gateway that supports different radio technologies, via gateways that talk to each other, or via an Internet service. Or two Internet services can talk together. But manufacturers collaborating to make their devices work together is not quite as open as true interoperability.

Interoperability can be improved through the use of web standards, but there are still many issues to overcome, such as ensuring devices remain secure.

Users will expect devices to interoperate. But right now, onus is on the user to research which devices will talk to each other, and it can be hard work to get them to coordinate.

Platforms designed to support a wide range of devices may not offer full support for the functionality of each individual device. They may of necessity have rather generic UIs designed to support multiple devices of the same type. While these support interoperability, this can come at the cost of a compromised UX. And while it may seem superficially appealing to control many devices through a single app, this may not scale.

Currently, closed ecosystems are likely to offer a more mature consumer experience, but in the long run interoperability gives users more control and freedom.

## [Case Study 4]

# LOOP: Connected Pelvic Floor Exerciser

**BY ADAM SCHEURING**

LOOP is a smartphone-connected, interactive pelvic floor muscle exerciser for women. It's a modern, designful take on a medical device used in physical therapy practice for a long time. The story of its invention started in February 2013 when a company called Thalmic Labs released the Myo armband, a human-computer interface device that presented a consumer-friendly wearable implementation of a commonly used medical sensor technology called EMG (electromyogram). This sparked our imagination and we started further researching EMG, which then led to the discovery of various vaginal and rectal muscle sensors and stimulators used by physical therapists and doctors for the treatment of various medical conditions related to the pelvic floor muscle group, such as stress urinary incontinence. These devices were rigid cylindrical objects with metal pads carrying bio-signals from the most intimate parts of the human body to clunky white-pink plastic boxes with a low-contrast \$1 LCD display (Figure CS4-1). In the age of iDevices, this lack of design appeal was shocking and sparked the idea to reinvent this product.

Our team set out to do this using our expertise in design and technology. Half of the team (Anders and Adam) previously worked as engineers at a semiconductor company making low-energy microcontrollers and Bluetooth radios. Many of the customers using these chips were developing fitness trackers, wearables, and digital health products. The other half of the team (Karianne and Line) worked for more than six years on designing high-end ergonomic sexual wellness products, and therefore were well acquainted with creating products that work well for the female body. The main idea was to create a better

pelvic floor muscle exerciser for women by applying the technology and the concepts of behavior change used in quantified self products like Fitbit combined with the design principles and high-quality, body-safe silicone materials used in high-end sexual wellness products.



**FIGURE CS4-1**

Most pelvic floor exercisers look like medical devices (images: Savantini, Laborie, Aquaflex)

To better understand the entire problem space, we started researching the medical issues related to weak pelvic floor muscles. Although the researchers' opinions on how to evaluate the self-reported data from patients are not completely aligned, the prevalence of these issues is surprisingly high. Studies estimate that 75% of women have some issues related to pelvic floor weakness throughout life. More than 30% of women suffer from some form of incontinence and the majority of them from stress urinary incontinence, the involuntary leakage correlated with coughing, sneezing, or other physical effort. We found stories of women in their early 50s who stop going out with their friends and become socially isolated because of the fear of leaking when laughing or sneezing. We found that more than 50% of those suffering from incontinence never actually talked to a physician, and many don't even talk to their spouses about the problem. In addition, people often described their experience with the existing medical products as "undignified and embarrassing." While incontinence pads comprise a \$7 billion global business, the adverse effects of the disease

cost healthcare systems around the world even more. Although many women suffer from sexual dysfunctions that are not broadly researched in medicine, they remain mostly the discourse of psychology and not mainstream population health. The subject is too touchy for our culture. The good news is that science also showed that exercising the pelvic floor muscles regularly can prevent or diminish stress incontinence and has positive sexual health effects for women of all ages. Yet many women don't know about it and even if they do, they don't do these exercises. We felt that we stumbled upon a big problem that has little awareness and poor existing solutions. We decided to create a product that is positioned in the realm of health and fitness and removes the negative connotations from both the adult toy and the medical device industry (see Figure CS4-2).



**FIGURE CS4-2**

The LOOP device and app

We identified two main problems with related pelvic floor muscle exercises. People were asking:

*“How do I know that I’m doing it right?”*

Lack of feedback of the correct muscle contraction

*“How do I know that I’m getting stronger?”*

Lack of strength tracking, and therefore lack of motivation over time

We knew that we needed a solution with a feedback loop.

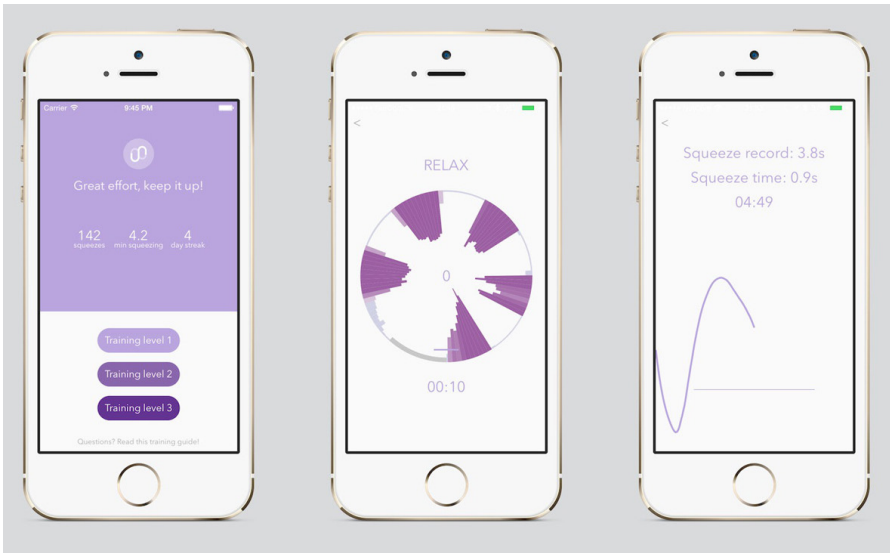
We started sketching the design and made working prototypes within a few months during the summer of 2013. We started researching medical papers, and talked with physical therapists, gynecologists, midwives, yoga and Pilates instructors, new mothers, and women of all ages. We made a basic prototype: a purple pillow made out of rubbery plastic material containing a circuit with a Bluetooth LE chip and a pressure sensor. We made a very simple app that was demonstrating the concept and the accuracy of the sensor. It wasn't body safe, therefore not testable. But it was good enough for starting discussions around the idea.

Based on this research, we made a few key observations early on:

- Pelvic floor muscle exercises work well and people don't like electric stimulators.
- Wireless changes the game because it allows people to do the exercises in various places. No need to be locked up in the bathroom anymore while doing the exercises.
- People want to know if they are doing the exercises correctly: 1 out of 3 women can't contract their pelvic floor muscles correctly without feedback. Accurate, real-time visual feedback is important because the muscles are a hidden part of the body.
- Motivation is key. Simple metrics, reminders, and accurate tracking of strength improvements go a long way to facilitate the behavior change.
- Peer support and coaching are effective forms of maintained motivation.
- Charging is a terrible user experience for intimate products. Remembering to keep them out of sight (e.g., hide them from their kids) puts extra mental load on people.
- Button switches for power and Bluetooth pairing are difficult for less dexterous people.
- Most people wanted the experience of the "weight-scale" and the "personal trainer." The use case for a wearable, always on sensor didn't seem very compelling.
- Muscle strength measurements are hard to reproduce.

- Medical professionals (physical therapists, gynecologists) were reluctant to recommend legacy medical devices that are aesthetically unappealing.

As the next step, we created a body-safe prototype and a simple smartphone app with only one screen that featured a line going up and down along the horizontal axis based on the force exerted by the pelvic floor muscles (see Figure CS4-3). We recruited a handful of testers who were willing to give us early feedback about the prototype of such an intimate product. We asked them to insert the body-safe device in the vagina, try to contract the pelvic floor muscles, and report their experiences in a one-on-one interview with one of our female cofounders. After this first testing round, we figured that people found the visualization itself quite motivating and informative. What we identified from the early feedback was that people would have benefited from seeing how consecutive contractions measured up against each other because this allowed them to quickly improve their contractions as part of the exercise. They also wanted simple statistic and timing along with instructions on when to contract the muscles and for how long.



**FIGURE CS4-3**  
LOOP app screens

We added these features and organized another round of test. This time we gathered more than a dozen testers and split them into groups. One group continued with giving feedback in one-on-one interviews, one group provided feedback through a form, and a few people allowed our female cofounders to observe them while they were using the product. We confirmed that the circular visualization allowing users to see consecutive contractions worked well. But we were surprised to see that the guiding line moving up and down designed to give instructions to the user on when to contract and relax confused some people.

In the next iteration, we will add a vibrating buzz on the phone to trigger the user to contract the pelvic floor muscles and we will simply use an interactive text as an instruction instead of a moving line. Some testers complained that our white background made their phones light so brightly that it disturbed their partner when they tried the product in bed at night. So we decided to introduce a night mode.

People really wanted a fitness/wellness product that is fun to use, that doesn't make them feel sick but doesn't look like a common sex-toy either. They were validating our hypothesis that this product fits into a new product category within health and fitness.

We wanted to create a fitness product that is visually appealing and fits into the daily life of women. LOOP has no wires or buttons, and does not require charging. It's simple and ergonomic, and the sensor is accurate and runs on a replaceable coin cell battery for up to one year. It has an accelerometer that detects posture to make it easier to reproduce measurements. People perceived the app as clean, easy to use, and motivating.

We received great feedback from testers and healthcare professionals. We believe that LOOP is more than just a tool. It's a new way of thinking and talking about the health aspects of one of the most important parts of the female body. We are proud to work on a meaningful cause even if it means that we have to challenge social norms and fight prudery, but we'll keep winning those battles. This is our contribution to a healthier, happier future.



# Responsible IoT Design

**BY ANN LIGHT AND CLAIRE ROWLAND<sup>1</sup>**

This chapter looks at the factors that make products trustworthy and safe to use. Connecting up products and services brings new challenges for security, privacy, social engineering, and the environment. We argue that we should consider how our designs impact on others' lives, and take this as seriously as we take the need for profit and competitive advantage.

This chapter introduces:

- The challenges of securing the Internet of Things (see page 420)
- The challenges of protecting users' privacy (see page 436)
- The risk that designing technology involves social engineering (see page 450)
- The environmental impact of IoT (see page 457)

This chapter addresses the following issues:

- Why embedded devices are especially at risk from network security threats (see page 425)
- What design requirements for usable IoT security might look like (see page 429)
- How data that seems innocuous can be aggregated to create a privacy breach (see page 442)
- What good data protection for IoT might look like (see page 445)

---

<sup>1</sup> Thanks to Chris Palmer.

- How technology that tries to be “smart” can end up reinforcing social stereotypes (see page 450)
- Understanding the environmental impact of a connected device across its entire lifecycle (see page 459)

## Security

Computer security is the degree to which a system can protect the assets it contains from unauthorized access, modification, or destruction. The “CIA triad” is a commonly used model of IT security, which defines three important aspects:

### *Confidentiality*

Protecting information from unauthorized access—for example, through eavesdropping or spoofing (masquerading as a legitimate user). (This is related to privacy, which we’ll discuss later.)

### *Integrity*

Protecting information from being modified or deleted by unauthorized parties (e.g., tampering with data or introducing viruses).

### *Availability*

Protecting the system from threats that would stop it working properly, resulting in loss of access to resources (e.g., loss of power, hardware failure, or a denial of service attack).

In UX terms, a “secure” system gives the user confidence that they know who they are dealing with and that information will reach only the intended target. They can trust that interactions with the system will be honest, do them no harm, and not result in any hidden consequences.<sup>2</sup>

Systems can never really be assumed to be 100% “secure” or “insecure.” Effective security is a question of risk management. Determining how much and what type of security is appropriate involves considering the value of the assets, the impact of their potential loss, and the potential vulnerabilities and threats. A public air quality sensor sharing data

---

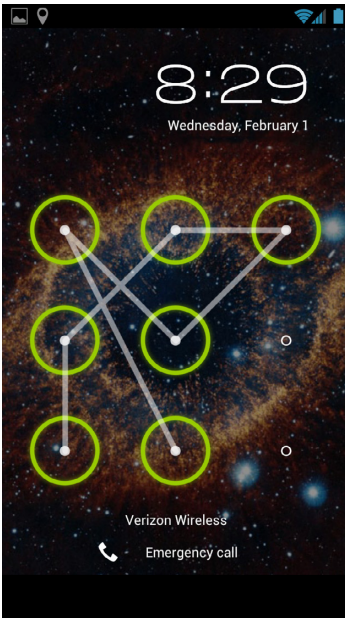
2 John Karat, Clare-Marie Karat, and Carolyn Brodie, “Privacy, Security, and Trust: Human–Computer Interaction Challenges and Opportunities at Their Intersection,” in *Human–Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition*, ed. Julie Jacko (Boca Raton, FL: CRC Press, 2012).

over the open Web may not have any means to control who can access its data. But it is not “insecure” if there is no way to cause harm (e.g., as long as faking the data from the sensor has no dangerous consequences; we’ll discuss this further momentarily).

The most important defense mechanism for a system that interacts with humans is generally *authentication*. This means ensuring that the system and user can be confident of each other’s identity. This is often done through a shared secret. Shared secrets are often characterized as:

- *Something I know* (e.g., text or graphical password; see Figure 11-1)
- *Something I have* (e.g., smartcard or physical key; see Figure 11-2)
- *Something I am* (e.g., biometrics like fingerprints or iris recognition; see Figure 11-3).

The first two of these place more cognitive load on the user: remembering a password or remembering to carry a key. The latter, in theory, does not.



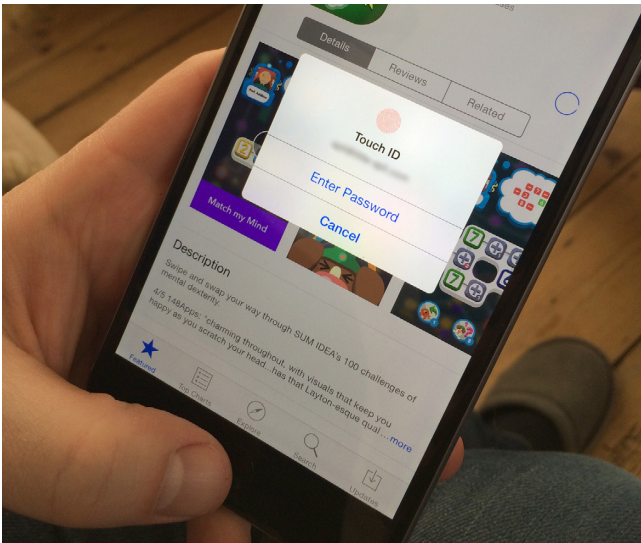
**FIGURE 11-1**

The Android unlock pattern is an example of a graphical password, or “something I know” authentication (image: Brent Rubell)



**FIGURE 11-2**

A simple house key is a basic form of “something I have” authentication



**FIGURE 11-3**

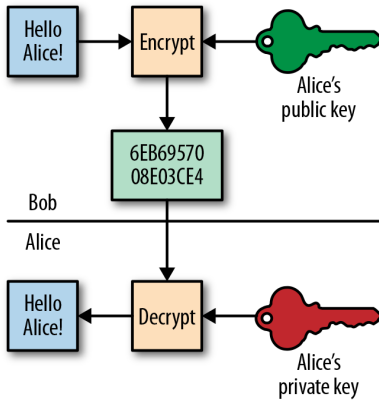
Apple TouchID fingerprint sensing is a form of “something I am” authentication

Authentication can also be required when two systems communicate without any human intervention.

Following on from authentication is *authorization*. A user (or another system) may be permitted to access certain data or functionality on the system, but not others.

*Encryption* disguises information to keep it secret, on devices, servers, or in transit over the network. Network security methods often aim to verify that information has not been observed or modified in transit (“man in the middle” attacks). A common approach is *public*

*key encryption* (see Figure 11-4). Each party has a public software key, which is published, and an associated private key known only to them. To ensure confidentiality, the sender encrypts the message using the recipient's public key. It can only be decrypted by the recipient, using their private key. The message sender also “signs” the message using their own private key. If the signature can be authenticated by the recipient using the sender's public key, then the message has not been interfered with in transit.



**FIGURE 11-4**

Diagram of public key encryption

Good security is a building block for privacy. If a security breach results in an unauthorized party gaining access to information, it can cause a privacy breach. But not all privacy breaches are security breaches. Parties who have legitimate access to data can cause privacy breaches by sharing it inappropriately with unauthorized third parties. These are failures to protect privacy rather than security breaches.

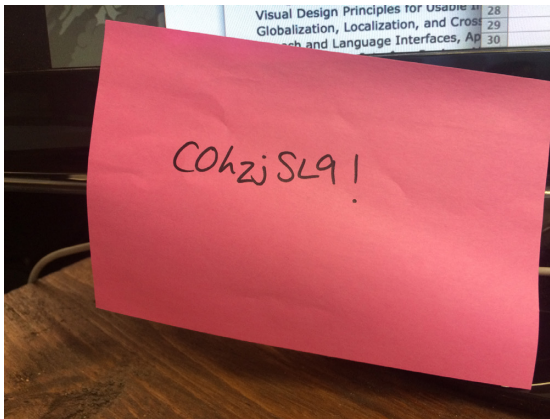
## THE UX OF SECURITY

Security and usability are often in tension. Usability seeks to make it *easy* for people to do things. Security measures often make it *harder* to do things. Both sides want to enable the *right* people to do things. But keeping the wrong people out adds friction to the process of getting

those things done. Security is not the user's main goal. Researcher Frank Stajano describes security as a "tax on the honest" caused by the existence of the dishonest.<sup>3</sup>

Computer security is often focused on minimizing the risk of malicious attacks to stored data or messages in transit. These elements don't always have a direct impact on the UX design. The biggest UX impact is usually around trade-offs between rigorous authentication and ease of interaction.

One of the big challenges in security is getting people to use it correctly. Security measures are often complex, often designed with highly trained users in mind. The more complex they are, the less likely they are to be used properly. Sophisticated security measures used badly (because they are too confusing) are often less secure in practice than less sophisticated, more usable measures used properly. For example, truly random passwords are harder to break. But they are unmemorable to users, who write them down on notes stuck to computer monitors, or left in unlocked drawers (see Figure 11-5).



**FIGURE 11-5**  
A random password stuck on a monitor

3 Stajano, Frank. "Security Issues in Ubiquitous Computing." University of Cambridge Computer Laboratory, Cambridge, UK (Jan 2009). <http://www.cl.cam.ac.uk/~fms27/papers/2008-Stajano-ubiquitous-DRAFT-80.pdf>.

It's not that users don't care about security. No one wants their bank account to be hacked. But nonexpert users often don't understand how things may be at risk, nor what to do to protect them. Witness the huge number of improperly secured home WiFi routers (see Figure 11-6). Nor can they foresee how their actions (such as sticking passwords on monitors) may increase risk. And if the tools are too much effort, they will try to find ways around them.



**FIGURE 11-6**

Security expert Gene Bransfield created the WarKittteh cat collar, which maps neighbors' WiFi networks as the cat prowls the neighborhood. The collar was a Black Hat security conference demo rather than a real threat but exposed a surprising number of open and poorly secured WiFi routers<sup>4</sup> (image: Gene Bransfield).

## WHY IOT SECURITY IS A BIG CHALLENGE

*If IoT is as flagrantly unsafe as our current most-IoT-like ecosystem (i.e., home WiFi routers), IoT will have fatal trustworthiness problems and might take off and then suddenly crash as an industry.*

**SECURITY ENGINEER CHRIS PALMER<sup>5</sup>**

Securing the Internet of Things is more challenging than securing the conventional Internet.

---

4 <http://www.wired.com/2014/08/how-to-use-your-cat-to-hack-your-neighbors-wi-fi/>

5 Personal communication. Palmer blogs at <https://noncombatant.org/>.

Connecting up the physical world creates the potential for malicious hacking to have “real world” consequences. In 2013, malicious hackers accessed a video baby monitor and verbally abused the child over the loudspeaker.<sup>6</sup> A security breach could easily put lives in danger. Cars could be remotely hijacked to disable the brakes. Pacemakers could be compromised to kill cardiac patients. And public infrastructure, like the water supply or transit system, could be hijacked to cause illness or disruption to millions:

The smart, connected fridge that will know when I've run out of milk and automatically place an order seems like a benign addition to my house. But when that fridge also has access to my credit card and can wirelessly unlock a door for a delivery person, it becomes less benign, especially if it depends on a security model designed for a fridge that only plugs into the power outlet.<sup>7</sup>

#### **RISK MANAGEMENT CONSULTANT CHRIS CLEARFIELD**

Securing the sheer number of additional things that are newly at risk will require huge effort. The patterns we have for authenticating users over the conventional Internet may not scale. Who wants to log into their thermostat every time they want to adjust the heating?

Embedded devices themselves will often be particularly vulnerable. Low margins on hardware mean that devices are often made as cheaply as possible. Many were not historically designed with security in mind, because they simply weren't at risk. Manufacturers may find it challenging (and expensive) to rise to the threats that come with Internet connectivity. Software installed during manufacturing may already be out of date by the time the device is purchased, and may have known vulnerabilities. As discussed in Chapter 2, many connected devices, from PVRs to smart watches to fridges, run general operating systems (often variations of Linux). This makes them vulnerable to the same, normally well-documented, exploits that work on laptops or phones.

---

6 <http://www.cnbc.com/id/100965217>

7 <http://blogs.hbr.org/2013/06/rethinking-security-for-the-in/>



Even if it's possible for the user to update the security settings or software, they may be unaware that the device is at risk, unsure what to do, or they may simply not bother. Most people have a million more pressing or interesting things to do than installing security patches on tens of household devices. And they are very likely to accept default settings when setting up a device, making it very important that those defaults encourage good security choices. But this often isn't the case.

In 2014, the website *insecam.cc* (now defunct) made available thousands of live feeds from home security cameras and video baby monitors. The site's creator claimed this was done to highlight the poor security of the many cameras, which by default either did not require user authentication, or on which the default password had not been changed. The camera owners were completely unaware their daily lives were being broadcast around the world.<sup>8</sup> In an even more troubling example, researchers at the Department of Homeland Security identified 300 medical devices with unchangeable passwords, which could allow someone to log in and change critical settings.<sup>9</sup> This raises the possibility that patients' lives could be put at risk by compromised pacemakers, defibrillators, and insulin pumps.<sup>10</sup>

In 2013, security researchers at Trustwave discovered a vulnerability in a (now-discontinued) Insteon home automation product that allowed home controls to be indexed by web search engines. As the system (by default) did not require authentication, users' lights and appliances could be controlled online by anyone.<sup>11</sup> This is an example of the kind of issue that can occur when Internet connectivity is bolted on to a product that was previously unconnected and thus not at risk.

The same researchers also identified that the Android app used to control a \$4,000 smart toilet had a hardcoded Bluetooth PIN. This meant that anyone with the app could control any of the manufacturer's toilets as long as they were within Bluetooth range. Those with a malicious

---

8 [http://www.huffingtonpost.co.uk/2014/11/24/insecam-shut-down\\_n\\_6212156.html](http://www.huffingtonpost.co.uk/2014/11/24/insecam-shut-down_n_6212156.html)

9 <http://bit.ly/1CNE7YX>

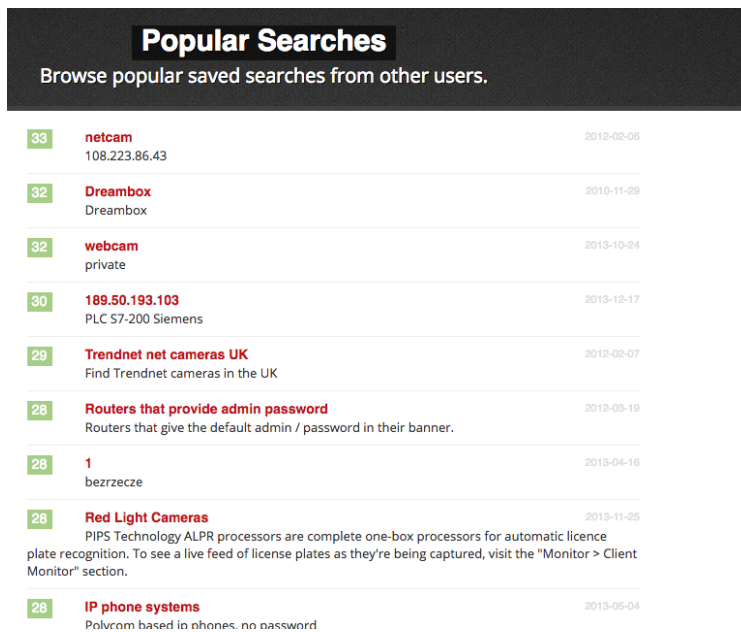
10 It is arguably worse to allow the user to secure a system with a default username and password than not to require any authentication at all, as the user is likely to believe the system is secure when it isn't. If no username and password is required, the user is at least aware the system is unprotected.

11 <http://www.forbes.com/sites/kashmirhill/2013/07/26/smart-homes-hack/>

bent could stand outside the bathroom while someone else was in there, raising and lowering the lid, flushing, and activating bidet and air-dry functions.

In the case of some commonly sold products, anyone who can get onto the local WiFi network is assumed to be an authorized user. Poorly secured WiFi can therefore be an easy route to compromise a system. Smart plugs are one example: a plug attached to a heater or hair straighteners could potentially be used to start a fire.

Even industrial systems are often poorly secured. Shodan,<sup>12</sup> a search engine for connected devices, has been shown to be able to find many consumer and industrial devices with minimal security. Devices listed include traffic lights, security cameras, home heating systems—even nuclear power plants and particle accelerators (see Figure 11-7). In many cases, devices could be accessed using the username “admin” and password “1234”.



**FIGURE 11-7**

Popular Shodan searches include red light cameras with unencrypted feeds (image: shodan.io)

<sup>12</sup> <https://www.shodan.io/>

Some very basic, low-powered devices, like burglar alarm or temperature sensors, may not have enough onboard computing power to handle encryption. They transfer data “in the clear,” so are prone to eavesdropping attacks.

Complex interconnected networks of devices will provide many opportunities for malware to spread. Low-powered devices with less robust security might provide an entry point for criminals looking to compromise more powerful devices in the system.

Devices with no UI, or very limited UIs, may be unable to tell us when they have been compromised. A recent botnet attack co-opted 100,000 devices including WiFi routers, connected TVs, and at least one connected fridge into sending out hundreds of thousands of spam emails.<sup>13</sup> How is a fridge going to tell you it’s been hacked?

All of these factors amplify existing problems creating usable security. There are no easy answers to the problem of good security for connected devices, especially those that will be used by nontechnical users. IoT systems are comprised of physical devices, users, and web services, and any one of these can be compromised, either locally or over the network.

In the next section we’ll consider what some design requirements for usable IoT security might look like.

## **DESIGN REQUIREMENTS FOR USABLE IOT SECURITY**

Building devices with more robust security models and securing network transmissions will solve some of these issues. But as far as the user’s concerned, these improvements aren’t visible in the system UX. In this section, we’ll focus on aspects of security that have a more direct impact on design.

To secure a conventional computer, a user might set up a login, install anti-virus software and keep it up to date. That’s just one device, but it already feels like an effort to many of us. IoT security is more challenging because it concerns systems. Functionality is distributed across networks of devices and web services. Not only do users need to authenticate themselves to the system and be authorized to do things, but

---

13 <http://www.proofpoint.com/about-us/press-releases/01162014.php>

devices and web services need to be able to act on behalf of the user. In a system of any complexity, it's not practical for them to require our explicit permission for every action or type of action. Nor do we want to wade through a huge list of checkboxes to pre-approve what they can or cannot do. We need to remain safe and in control, but we also need to be able to trust that systems will employ sensible defaults and grant some of our devices a degree of autonomy.

The most secure approach is not to connect a device to anything at all. There's a lot to be said for not connecting things (either to each other or the Internet) without a strong case for doing so. But it would be a shame for IoT to be unnecessarily limited in this way.

### **Limit the damage that can be caused**

Security will be breached at some point, so damage limitation is important. Devices should, where possible, be designed to do no harm when compromised. But this may not always be possible. A car is of no value unless it can drive at reasonable speed, and at that speed it could cause death or injury in a collision. The car could disable itself if it knows it has been hacked, but malware is often designed to ensure the host device is unaware it has a problem.

Limiting devices to a very specific set of functionality will help prevent them being roped into unintended activities. For example, a WiFi-connected oven could be co-opted into sending spam email, sniffing the home WiFi network for sensitive details like bank account details, or compromising its functionality to endanger the user or environment (e.g., turning on the gas to cause a leak). It is safest to build it in such a way that it cannot support some of these behaviors.

A breach in one area should not enable malware or criminals to take over control of the entire system. For example, breaching the in-car entertainment system should not be a back door route to accessing the navigation system or anti-lock brakes.

### **Keep devices secure**

Security software needs to be kept up to date, ideally with as little user intervention as possible. This can be complicated to deliver when there are multiple business partners involved in supporting a system. For example, most current Android smartphones run out-of-date software

and often have security vulnerabilities. Google provides software updates to fix these issues but they are not packaged up and pushed to consumers by the carriers or handset manufacturers.<sup>14</sup>

Security engineer Chris Palmer proposes that if devices can't be updated and maintained, they may need to take themselves out of service when they get too old, or at least disable any security-critical functionality.<sup>15</sup> Of course, devices shutting themselves down or shutting off functionality could also cause problems. The appropriate solution will vary across different devices, applications, and domains. Keeping devices in use longer through frequent software updates is more environmentally sound. But to date, the service infrastructure tends not to support this.

The physical security of devices is also an issue. If possible, product and system designers should take steps to make them less likely to be stolen or physically accessed by unauthorized parties, such as designing product housings to prevent tampering or make it apparent when the device has been tampered with (see Figure 11-8).



**FIGURE 11-8**

A hardware cryptographic security module encased in a tamper-proof housing (image: Mils Electronic)

A gateway can provide the security necessary to protect a large number of low-powered edge devices. It's also easier to maintain one firewall for many different devices than to have to worry about securing them all individually. But any device that is open to the Internet may have vulnerabilities that could be targeted.

<sup>14</sup> <http://bit.ly/1CNFHdw>

<sup>15</sup> Personal communication.

## Make authentication easier

Users may need to authenticate themselves to devices and services. Devices and web services may also need to identify themselves to other devices and web services.

User authentication often involves passwords (“something I know”), but entering passwords to interact with many different devices would be onerous. In some cases, possession of a phone or smartcard (“something I have”) may be enough (e.g., my phone can control my home lights because it is my phone). But devices can fall into the wrong hands. Phones, keys, and smartcards can be stolen, copied, or cloned. Biometrics, a form of “something I am” (e.g., the iPhone’s fingerprint recognition, iris recognition, or even heart rate or gait analysis), can be low-friction methods of user authentication. But they are not always reliable and can increase the personal risk to users. In 2005, a violent gang of carjackers in Malaysia chopped off a victim’s finger to get around the car’s fingerprint recognition system.<sup>16</sup> Using more than one channel or method (e.g., Google’s two factor authentication, see Figure 11-9) is extra secure, but more hassle.



### Two-step verification

Enter the verification code generated by your mobile application.

Enter code:

Remember verification for this computer.

[Get a new verification code](#)

**FIGURE 11-9**

Google’s two factor authentication requires the user to use a password and a one-time code sent to their mobile via SMS (image: google.com)

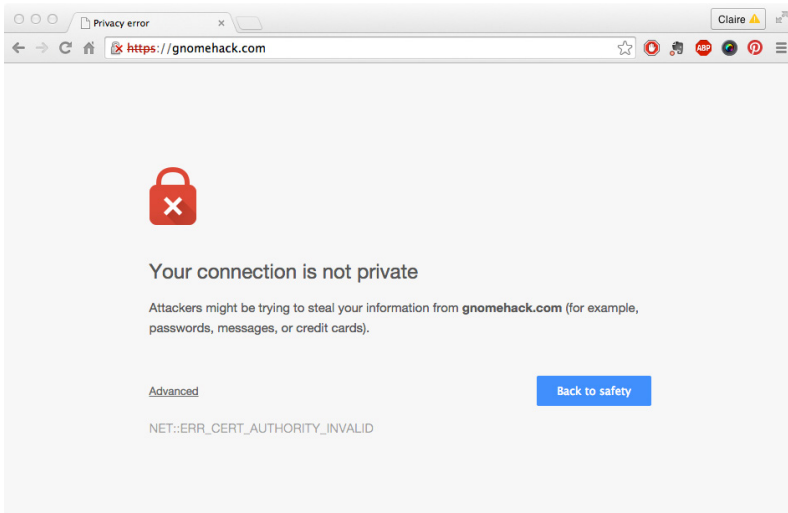
In order to work together, devices need to know which other devices they can trust. Your car needs to prove that it is genuinely your car as it unlocks the garage door. You can verify the identity of the device in two ways: either ask the user to do it, or ask an independent, trusted authority to do it. You might need verification every time, or just the first time.

---

16 <http://news.bbc.co.uk/1/hi/world/asia-pacific/4396831.stm>

As Chris Palmer points out,<sup>17</sup> requiring the user’s agreement every time won’t be practical across more than a handful of devices. So devices have to find ways to validate each other’s identity and communicate securely. On the Web, the identity of an entity with a public name like `mail.google.com` can be authenticated through a certificate associated with its DNS entry. If your browser cannot find a certificate proving that the website claiming to be `mail.google.com` really is `mail.google.com`, it will flag it up as a security risk (see Figure 11-10). But users often simply click past such warnings.

But many IoT devices may not have public names and DNS entries, so cannot currently be independently validated in the same way. How does your car prove it really is *your* car and not a spoof device just pretending to be your car? There are currently no standards here. If some kind of third-party verification for connected devices were available, the garage door could request the car’s “papers” the first time it sees the car, then just trust it in future. Or it could ask the user for authorization on first use, then just connect in future.



**FIGURE 11-10**

A browser certificate warning

<sup>17</sup> Personal communication; see also his blog <http://noncombatant.org/2014/10/12/brainstorming-security-for-the-internet-of-things/>.

How do we get humans involved in making these decisions without blindly accepting any request from a device wanting to connect, or clicking past any irritating warning that a device's certificate may not be what it seems? This is a question that still troubles browser designers and won't get any easier the more devices begin to clamor for our attention.

### **Keep users in control of permissions**

Authenticating devices is one step, but users also have to be in control of how much those devices can do or what they can share. The clichéd Internet fridge may have your supermarket account and credit card details so it can order more milk and eggs. But it mustn't be allowed to use those details to buy anything else that you haven't approved. Permission control might mean devices and services have only partial access to features on other devices and services, or are granted access only for a limited time. Or a device can be bonded as a "slave" to another device and can only be controlled by it, as a smart TV might be linked only to one smartphone or tablet.

Users should of course be in control, but people don't want to tick lots of boxes every time they get a new device to specify what it can and can't do. Some combination of user control and low friction, sensible defaults is needed: a trade-off of transparency and effort. One possible model might be for users to treat devices like people they have hired to work for them—for example, granting them the kind of agency and trust you might give a housekeeper. Academic and commercial researchers are working on shared standards ways of enabling user authentication and access permission setting for connected devices, but there are no widely used solutions yet.

Authentication and authorization can to some extent be automated and happen without user interaction. But this doesn't work if the user needs to determine what level of security is appropriate or how to use information appropriately.

### **Make the invisible visible**

Devices with limited UIs can't always tell us when they have been compromised. Is your dishwasher trying to tell you it's been co-opted into a botnet or does it just need more rinse aid? Ways of making these invisible threats visible are needed, which requires both a way of detecting



and informing the user of these threats. For example, software agents might monitor system activity and inform the user about potential security threats and available countermeasures.

## Design security measures to suit user and domain needs

Security measures must be appropriate to user needs. There is no point requiring consumers to do something they are not capable of. Although not an IoT example, the One Laptop Per Child security principles are an interesting example here. These establish that the security measures on the computer must be suitable for users who cannot yet read, or choose and remember a password (see Figure 11-11).

This document is complete with regard to the OLPC security model, but is generally non-technical. A separate document is being prepared that complements this one with fully technical descriptions and commentary.

### Principles and goals

---

#### Principles

##### Open design

The laptop's security must not depend upon a secret design implemented in hardware or software.

##### No lockdown

Though in their default settings, the laptop's security systems may impose various prohibitions on the user's actions, there must exist a way for these security systems to be disabled. When that is the case, the machine will grant the user complete control.

##### No reading required

Security cannot depend upon the user's ability to read a message from the computer and act in an informed and sensible manner. While disabling a particular security mechanism *may* require reading, a machine must be secure out of the factory if given to a user who cannot yet read.

##### Unobtrusive security

Whenever possible, the security on the machines must be behind the scenes, making its presence known only through subtle visual or audio cues, and never getting in the user's way. Whenever in conflict with slight user convenience, strong unobtrusive security is to take precedence, though utmost care must be taken to ensure such allowances do not seriously or conspicuously reduce the usability of the machines. As an example, if a program is found attempting to violate a security setting, the user will not be prompted to permit the action; the action will simply be denied. If the user wishes to grant permission for such an action, she can do so through the graphical security center interface.

#### Goals

##### No user passwords

With users as young as 5 years old, the security of the laptop cannot depend on the user's ability to remember a password. Users cannot be expected to choose passwords when they first receive computers.

##### No unencrypted authentication

Authentication of laptops or users will not depend upon identifiers that are sent unencrypted over the network. This means no cleartext passwords of any kind will be used in any OLPC protocol and Ethernet MAC addresses will never be used for authentication.

##### Out-of-the-box security

The laptop should be both usable and secure out-of-the-box, without the need to download security updates when at all possible.

##### Limited institutional PKI

The laptop will be supplied with public keys from OLPC and the country or regional authority (e.g. the ministry or department of education), but these keys will not be used to validate the identity of laptop users. The sole purpose of these keys will be to verify the integrity of bundled software and content. Users will be identified through an organically-grown PKI [\[1\]](#) without a certified chain of trust—in other words, our approach to PKI [\[2\]](#) is KCM, or key continuity management.

##### No permanent data loss

Information on the laptop will be replicated to some centralized storage place so that the student can recover it in the event that the laptop is lost, stolen or destroyed.

## FIGURE 11-11

The One Laptop Per Child security principles (image: laptop.org)<sup>18</sup>

---

18 [http://wiki.laptop.org/go/OLPC\\_Bitfrost#Principles](http://wiki.laptop.org/go/OLPC_Bitfrost#Principles)

Different domains (e.g., healthcare, banking, and home) may have unique requirements. The nature of the system and data, the context in which they are used, and the consequences of them being compromised may be different in each case.

## Privacy

If security is a network issue, privacy is a networked data issue. It is about collecting information about other people, and making it available, as opposed to the simple exploitation of weaknesses in the system. Often privacy and security are interwoven, because a breach in security can result in a loss of privacy.

There is often no one right way to deal with privacy issues in design. Some data flows are essential to running a connected service, and with these, comes a flow of usable information that can be manipulated. It is not as easy as putting a lock on the door, which is essentially what securing your network does. Instead, it is about vetting who comes over the threshold and what they can access. Collecting only the minimum amount of personal data needed to deliver your service is an important first step.

Privacy issues are less predictable than security issues, as what counts as privacy is neither universally agreed, nor static. Privacy is dependent on contextual factors and local interpretation, as introduced in Chapter 6. In other words, one person's notion of a privacy violation is another person's opportunity to get a cheaper service. And privacy expectations change. At one point, there was a belief in some circles that privacy would be replaced by total transparency. Slogans such as "If you've got nothing to hide; you've got nothing to fear" implied that privacy was only an issue for the corrupt and the old-fashioned. However, since Edward Snowden's revelations of extensive NSA data monitoring, these sentiments have swung the other way. People are taking invasive practices more seriously.

Understanding the scope of privacy concerns helps in designing a system. It is possible to mitigate some of them, alert people where necessary, and generally choose options that make the flow of data more manageable. It is also worth noting that in some parts of the world, privacy legislation affects what it is possible to do.

## INFORMATION AND PRIVACY

One important part of privacy is about controlling information: what is released and who sees what. The services in and out of homes, cars, and health devices carry enough information to embarrass and compromise their users repeatedly. It is possible to make many accurate—and inaccurate—inferences about what each user or group of users is doing and why. And some of that information may find its way to places that users were not expecting, even if they allowed the collection of the data in the first place.

Once these streams of information and related deductions leave the hands of people permitted to know these details, a privacy violation has occurred. In some cases, this is trivial, as the breach involves nothing of any consequence. But different people have different ideas about what is compromising.

There are two types of privacy breach to consider.

First, there are streams of information leaving connected devices for other destinations related to the service provider or made accessible on the journey. These streams may pass through the hands of hackers, advertisers, partner services and companies, other third parties, the NSA, and so on. We will return to these later.

Second, there are the privacy breaches that the service commits locally among users sharing the same resources or space. As more data are stored, so there is more chance of revealing socially inappropriate details to other local users or doing something that makes them feel nervous. While these may or may not count as illegal violations, they do reduce the popularity of a design. We might call these breaches *tactless sharing*.

### Handling data with tact

The recent backlash against Google Glass has been tied to the insensitive handling of privacy in its design (among other things). Phone calls with a picture feed took more than 40 years to find their way into popular usage, partly because they threatened to give away too much about the person being phoned. (Video phone services that have taken off allow users to opt in or out of video. They are mostly used for pre-arranged calls, so that both ends can get their house in order, metaphorically or literally.) Now, wearables that record life and stream it are proving unpopular with the person on the other side of the camera. The

design of Glass does not make it clear if the wearer is using the camera to record at a particular moment or not. People have likened talking to someone wearing Google Glass with having a phone held to their face (see Figure 11-12). Even if the video is not in use, wearing the glasses sets up an asymmetric relationship that makes holding a conversation potentially compromising and certainly uncomfortable.



**FIGURE 11-12**

Google Glass can disconcert those on the other side of the camera (image: Loïc Le Meur, Wikimedia Commons, CC license)

Another form of invasion is even less obvious. Teenagers sharing a house with their parents and younger siblings may go to great lengths to hide what they are doing, even if it is perfectly innocent. Home technology might well reveal what they are up to as part of helping the homeowner organize household protocols. For the teenager, that is a breach of privacy just as sending dad's daily alcohol consumption figures to the local medical center would be, or inadvertently betraying an infidelity.

Similarly, there are ethical considerations that involve sharing data involving people considered vulnerable. Here are three sets of questions, all concerning different relations:

- Should parents be able to monitor their children freely? For example, see Figure 11-13. Does tracking the child's location or activity allow them more freedom (because parents trust them to be more

independent) or impinge on their freedom by denying them privacy? Until what age is it OK? Should parents seek their child's consent?



**FIGURE 11-13**

A child wearing an Amber Alert GPS tag (image: Amber Alert)

- At what point does it become acceptable to monitor your aging parent? Is counting beers, checking insulin levels, and locking the liquor cabinet a necessary precaution or an outrageous intrusion into someone's private life?
- What is the difference between an emergency alarm to be pressed by users if they feel ill and one that is wired to alert the local doctor if irregularities are detected by the monitor? And how is it different again if the aging parent is no longer well enough to give consent to surveillance?

The configuration of tools can leave these issues wide open, steer one way or the other, or give people the means to work out their own custom path.

Tact is required in considering local privacy for connected objects; a poor choice will affect take-up. By tact, we mean that what counts as intrusive is context-dependent, determined by who is present and what the situation is, so we need to err on the side of giving people discretion to go about their business, whatever that may mean to them. What counts as tactful in different cultures will vary. One way of finding

out what people think is to ask them (see Chapter 6). Bear in mind that people are particularly bad at anticipating privacy concerns. Ask a range of people about the specifics of their experience in different contexts. Consider tacit awareness of others and what happens when this becomes loudly signposted. It is best not to use hypothetical or generalized scenarios. And never assume that one person can speak for a whole household.

In 2004, the concept of “contextual integrity”<sup>19</sup> was introduced to draw attention to how images and the like can be compromised when they are viewed (and, of course, posted) out of context. It has been particularly useful to explain the discomfort that results when intimate pictures that were taken with consent for use privately are then circulated on the Web. It is easy to see that a similar violation is possible in the era of IoT, when data is used out of its original context.

### **Collecting and distributing data**

As the designer of a connected product, some information will be necessary to you to provide a good service. The more that tools and products are expected to infer users’ wishes from their behavior, the more information you will need to manipulate. Some of that information may also have to be given to third parties so that multiple services can interoperate. As products become more complicated, they will manage more and more information about people and their lifestyles, overseen by you as part of service provision. This brings responsibilities as well as opportunities with it.

At a recent conference discussing the future of IoT,<sup>20</sup> panelists noted that new sensing devices could be flagging up health issues before the person affected knows about them. They then got to arguing about how such insights could be used. One declared that any company not using its collected data to the full would be in danger of losing competitive advantage. Others were more optimistic about a self-regulating industry. There followed a disagreement about how ethical companies are likely to be, regardless of what is identified as best consumer practice.

---

19 Helen Nissenbaum, *Privacy as Contextual Integrity*, *Washington Law Review*, 2004.

20 Software AG’s *Innovation World* conference in New Orleans, Oct 2014.

Let's take their question further. A service provider that knows the user is ill before the user themselves does has various choices. They can tell the person; tell the doctor; tell the insurance company; or keep quiet. Each has different ethical implications.

Telling the individual may be awkward: What and how much do you say? Will they understand the implications of the diagnosis? Will it scare them more than help them? Will they understand how you know? Even a message telling them to visit their doctor is scary coming from a device.

Telling the doctor may overcome some of these problems, but telling the doctor is a breach of privacy unless it is an accepted part of the service. Should it be? Did you think you were building a medical aid when you made a system to help runners monitor their heart rate and blood flow? What does it say in the small print (that no one reads) about the limits of your responsibility? Is this the finite liability that your lawyers recommended? How might liability change as knowledge of medical conditions increases rapidly through such monitoring and correlation of conditions?

As for telling the insurance company, this might be lucrative for you as the service provider if you happen to have an arrangement or be owned by the same parent company as the insurer. But does the user know who will be informed of their health condition and what the implications are?

Different countries have different healthcare systems, not all insurance based. But health insurance companies across the world have an interest in health diagnostics tools. If they provide data to their clients to help them live longer and claim less, that offers a win-win situation. If everyone knows who is seeing what, there are likely to be fewer privacy implications. If the tool becomes a way of ensuring that the company is protected from risk, even if the client isn't, then users might think twice about a digital service that "leaked" such information. There would have to be some reward for parting with this highly personal data, especially if it resulted in higher premiums. Among other things, the challenge in each case is the right degree of transparency so that users can make an informed judgment in their interests.

The fridge with advertising, mentioned in Chapter 4, is a good example of how this trade-off might work. A free fridge that promoted food products to users based on what they eat would have to know a lot about them. It would be sending that information to the supermarket that issued the fridge. In this way, the supermarket would have millions of user profiles and could implement cost-effective just-in-time ordering, reduce wasted food and innovate with products that are sure to sell. Only people who are prepared to buy their ad-free fridge at full cost (see Chapter 4) would be buying real privacy about their food consumption. Otherwise, there would be a permanent spy in the kitchen. But it would be a known spy and one that could be informed as needed about changes to household life.

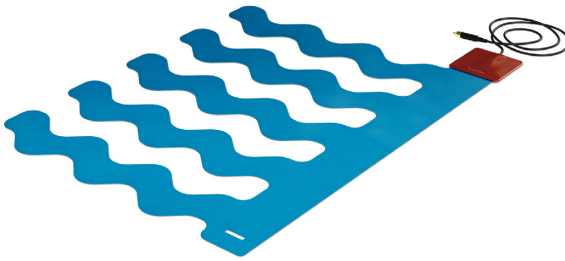
### **Aggregation**

One of the hardest things for people to understand is the way that innocuous information can be combined to give insight into beliefs, behavior, and choices. Connected product systems can use multiple data sources to make inferences about the world, especially where things are supposed to respond to context, hierarchies of user permissions, and contradictory inputs. For instance, it may be useful to the system to know which member of the family is opening the door at any moment. Keys can be customized to give this information, allowing a whole series of adaptations to take place based on other data held about each individual, from setting room temperature and TV channel to denying access to parts of the house. Just as our swipe cards at work hold this kind of differentiating information, so too can our homes. And this could be enhanced by network knowledge that the person at the door has had a difficult day, is running a fever, or wants to impress their date. The house might perform differently according to this knowledge, but handing over this information to the house is an act of trust that this won't be abused then or later.

Computer rules of great subtlety and complexity can be written to govern points of intersection between data types and make sense of them. In some fields, rules are being shown to outdo human judgment in aggregating many different variables to come up with a likely deduction, such as fraud detection and tumor identification. Seemingly innocent revelations over the years can result in unexpected (and sometimes unwelcome) insights, available to anyone holding multiple streams of data that can be combined and analyzed—and the power to aggregate them.



A classic story of inference that invades privacy is given by Professor Andrew Monk<sup>21</sup> in talking about care for older people. He describes research in a care home where elderly residents were asked if they minded having sensors attached to their beds (see Figure 11-14) and to the sinks in their bathrooms. These would record if they got out of bed for periods in the night and whether they had turned off the tap. In both cases, the residents were quite happy for this information to be gathered and given to the wardens in the home. The same residents were then asked if they would mind the wardens knowing if they got up in the night to use the toilet. Most were unhappy about the idea. They did not see that the combination of the two sensors would allow anyone scanning the data to infer this information. They had no sense that a simple program could scan for intersections in the data and conclude—without any human intervention—that they had been up for a pee.



**FIGURE 11-14**

An under-mattress sensor, such as this one made by Emfit, can be used to track whether the occupant has got out of bed (image: Emfit)

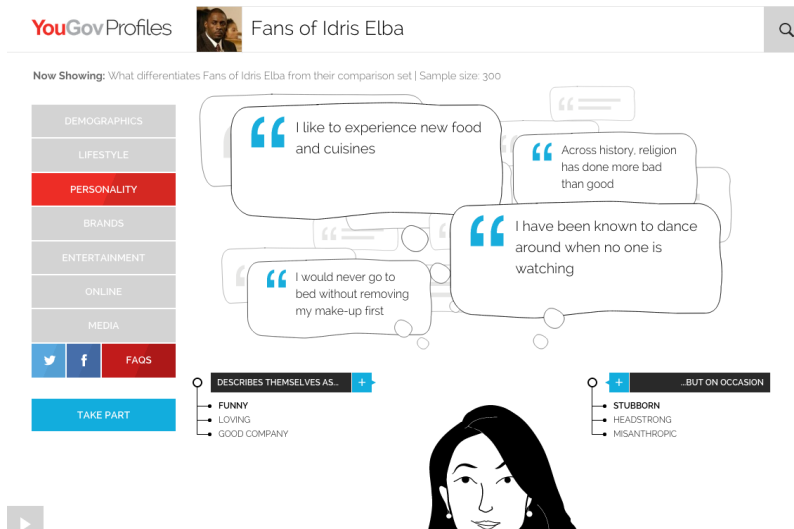
A more everyday example of data mining involves analysis of household purchases. A supermarket loyalty card can reveal family preferences and other superficial characteristics. Prolonged analysis over time can reveal if a baby is due, impending divorce, menstrual cycles, and other more intimate details. Added information from local devices, like notice of a fall in blood sugar, could be used to make inferences about the state of someone's mood, health, and/or hunger level at time of shopping and thus promote more effective selling. Once this is

---

<sup>21</sup> At the INTERACT 2005 conference, speaking on a panel on “Trust and Incidental Interaction: Would you let a Talking Paper Clip run YOUR Home?”

combined with household data from a smart home and medical, social, and financial data from other sources, a strong picture of each of us will be available to those who want to know about us. That will mostly be marketers (and possibly some government agencies). And they will have a very full grasp of our aggregated tendencies, which can be taken back to each of us as individuals.

The collective intelligence service Hunch.com, which ceased in 2014, showed just how much could be correlated, even without access to our home and health data. Hugo Liu, Hunch's chief scientist, could ask someone one question and then present them with a whole profile of tastes and behaviors. He once did this to members of a panel that Ann had co-convened and they were stunned by how accurately he could "interpret" them. It was a great party trick, but it made the point that competent cross-referenced inferences about our tastes and lifestyles already exist and all that is needed is to add sensors to know when and where to apply them. YouGov in the UK has a similar profiling service online that can build a rich set of demographics and preferences based around an interest in a single brand, person, or thing (see Figure 11-15).



**FIGURE 11-15**

Fans of the actor Idris Elba have an above-average propensity to dance when no one else is watching (image: YouGov)<sup>22</sup>

<sup>22</sup> <https://yougov.co.uk/profiler>

## **Patterns of behavior**

Privacy is mostly about information, though patterns of information flow are also relevant because these give access to patterns of behavior. A crude example would be turning off most of your network when you leave to go on holiday. Even the dip in energy consumption that your smart meter is picking up tells the story. That dip gives as much information as leaving a note for the milkman saying you don't need milk for the next week, as people used to do in more trusting times. It puts your home at risk by indicating you are absent, which is a security matter, but, taken over the year, it also reveals that you go away, when and for how long, how regularly you go, and so on. The only thing it doesn't indicate is where you have headed, but that is probably easy to establish by combining this data with recent buying patterns.

## **LEGAL ISSUES, CONSENT, AND DATA PROTECTION**

Different parts of the world take different attitudes to this data management responsibility. An American friend recently asked for credit to buy a large consumer electronics item. She spent several minutes on the phone to the technology company negotiating with them about the state of her finances. In Europe, the company would put in a call to a credit ratings company to get a rating for the individual and accept or deny her credit, but the firm itself would have less access to her financial information. In Europe, this information would be deemed private and legislation controls its distribution.

### **In the USA**

In the United States, there is no single, comprehensive national law regulating the collection and use of personal data. There is piecemeal legislation, and security breaches in recent years have led to the field of privacy becoming one of the fastest growing areas of legal regulation. But there are gaps, overlaps, and even contradictions in the many federal and state laws and self-regulated systems that are intended to protect individuals.

Security-related privacy laws exist that control use of information linking name, Social Security number, driver's details, home, telephone, email, and financial information, but only in some states. For instance, California has introduced the legal right for young people "to be forgotten" online. And recently a free mobile app that shared users' geo-location and unique device identification with advertisers without

providing notice or obtaining consent was prosecuted. The regulators are not without ambition, but there is too much complexity in the US situation to begin to cover it here.

## **In Europe**

By contrast, Europe has long had a centralized approach to data protection. At the time of writing, a new regulation is being introduced into the European Union (EU) that will supersede the current Data Protection Directive, active since 1995. In Europe, privacy is a human right and data protection is upheld more stringently than in much of the world.

Even passing information out of the EU is controlled, with a duty on service providers to ensure that the receiving country has similar protective legislation in place or contractual protections. This has implications for the siting of servers and so on. While online service providers have historically argued that resources are being reached from countries outside the EU (and so outside its jurisdiction), this is a more difficult argument to make when the data is associated with embodied objects that exist within the union's boundaries. The EU advises that any equipment located in an EU state is under the jurisdiction of European data protection laws and recent legal rulings have given individuals the right to be governed by the laws where they live.

Relatedly, the “right to be forgotten and to erasure” is being discussed in Europe at the moment. This is proving highly controversial. If and when enacted, it would allow people to request the removal of information that they deem compromising to the life they are living now—for example, information that is old or lapsed, or that pertains to a person's juvenile years.

## **Principles of data management for privacy**

The OECD, in 1980, produced seven principles that underpin the EU system. They are also endorsed in the United States, though not legislated for. The OECD principles are:

### *Notice*

Data subjects should be given notice when their data is being collected

### *Purpose*

Data should only be used for the purpose stated and not for any other purposes

### *Consent*

Data should not be disclosed without the data subject's consent

### *Security*

Collected data should be kept secure from any potential abuses

### *Disclosure*

Data subjects should be informed as to who is collecting their data

### *Access*

Data subjects should be allowed to access their data and make corrections to any inaccurate data

### *Accountability*

Data subjects should have a way to hold data collectors accountable for not following these principles

And Europe is leading again with IoT regulation. Already, there are extensive recommendations about what IoT services need to do to honor data protection, issued by the EU's Article 29 Working Party (and well summarized by lawyer Giulio Coraggio).<sup>23</sup> This includes making a privacy impact assessment of new IoT technologies. In Coraggio's view, service providers need to identify carefully roles and responsibilities in the processing of personal data by everyone involved in providing a service and the equipment to support it so that liabilities are well understood. He points out that any data—even if it originates from “things”—can be considered as personal data if it is able to reveal information about the personal life of individuals.

Showing the range of aspects that the working party has considered, two examples of the many new guidelines are:

- Information published by IoT devices on social platforms should not become public or be indexed by search engines by default
- IoT device owners and users should not be economically penalized or have degraded access to the capabilities of their devices if they decide not to provide consent

---

23 <http://www.gamingtechlaw.com/2014/09/internet-of-things-data-protection-2.html>

The second of these already has people scratching their heads. Implicit in this is the requirement to be able to handle data so that consent for release is not needed, as clearly some flow of information will be essential to provide “the capabilities of their devices” (i.e., services).

### **Privacy settings and informed consent**

There is no legal breach of privacy if consent for the use of the information in the way that it is being used has been given.

Twenty years ago, at the outset of ubiquitous computing, Victoria Bellotti and Abigail Sellen<sup>24</sup> of Xerox PARC argued for design capable of:

- Empowering people to stipulate what information they project and who can get hold of it (control)
- Informing people when and what information about them is being captured and to whom the information is being made available (feedback)

At the time, control still seemed plausible. Ethically, it is now debatable whether people are capable of giving *informed* consent in many of the cases we discuss here, as we have shown how hard it is to understand possible uses of data. It is also impossible to make absolute commitments for data that has been distributed to third parties. Both their plans and the plans of those to whom they may pass it are outside the designer’s control. It is similarly difficult to predict what data combinations might result from developments in a smart home as it takes in new nodes and features. Even a small device may have a stake in sharing this outside the boundaries of the original terms.

Put simply, getting informed consent is a minefield. It may be better to try and avoid the need for it.

### **“PRIVACY BY DESIGN AND PRIVACY BY DEFAULT”<sup>25</sup>**

One way of overcoming the problem of consent and data privacy is to store all data in the local system, where it belongs to the user, and only take the minimum of processed and anonymized information back

---

24 Victoria Bellotti and Abigail Sellen, “Design for Privacy in Ubiquitous Computing Environments,” *ACM CSCW* 93 1993: 13–17 .

25 A good overview of Privacy by Design techniques can be found in: Ian Brown, “Keeping Our Secrets? Designing Internet Technologies for the Public Good,” *European Human Rights Law Review* 4 (2014): 368–377.

into the service to provide necessary functionality. By anonymized, we mean data that cannot identify its source either in content or pattern. (It is no use just changing or deleting names if it is possible to work out who is being referred to by correlating with location, patterns of connection, etc.) In 2010, Privacy by Design was recognized as the global privacy standard by the International Conference of Data Protection and Privacy Commissioners. This can support tact in the moment, minimize long-term misuse of data, and reduce the potential for unpleasant repercussions after the event.

For instance, research is going on into dynamic privacy settings. “While driving to a client, Bob [the boss] may access [staff member] Alice’s destination and her estimated arrival time, but not Alice’s exact position. When Alice is at a client, Bob can know Alice’s exact location. At the same time, all incoming calls for Alice should be blocked in order to not disturb the meeting. After work, Bob should not get any information about Alice’s whereabouts.”<sup>26</sup> In this example, privacy and application blocking are linked, to “better motivate privacy relevance” and provide a more efficient meeting. But, importantly, this acknowledges that Alice’s role as employee has privacy-sensitive context features. It matters where she is and whether she is on duty or not. And the description recognizes that granularity changes (preciseness of location) affect her privacy.

Privacy at source sits in tension with some providers’ business models, which rely on gathering, aggregating, and trading in data. Bruce Sterling’s essay on “The Struggle for the Internet of Things”<sup>27</sup> relates how centralized IoT information could become if the big five super-companies he identifies start to manage every interaction users have with all their devices. If this is the future of the Internet of Things, then mass surveillance is a critical part of the plan. But even monetizing every interaction (as they may hope to do) does not require that precise knowledge of who is doing what is centrally stored and mined for individuals’ details, even while it is being analyzed for trends, relationships, and patterns.

---

26 Florian Schaub, Bastian Könings, Stefan Dietzel, Michael Weber, and Frank Kargl, “Privacy Context Model for Dynamic Privacy Adaptation in Ubiquitous Computing,” ACM UbiComp ’12, 2012.

27 Strelka Press, 2014.

It is perhaps optimistic or naïve to point out alternatives to expecting users to part with all their movements, actions, and desires to those with the resources to dig through this information. Certainly, there is unlikely to be any symmetry: government and big business will know about everyone's lives, but little information will come back the other way. However, there is a strong lobby, led by Canada, for de-identification of data before it is released. There are critiques of this system, such as the argument that data can be re-identified (and we cannot predict how clever applications for re-identification will become in the future). But on the whole, some thought to how much information you need to take—and in what form—is a good way to approach privacy issues. True (or “deep”) anonymization affords privacy. There is also a strong ethical argument for erasing data that is no longer necessary, rather than stockpiling it.

Such measures need to be implemented at design time and respond to context. For instance, the geographical location of devices (and their users) can be related to their current function in deciding what level of privacy is a basic right. And however data management is approached, a transparent and informative method of communicating with users is necessary. The current model of long, obscure End User Licensing Agreements (EULAs) that no one reads is insufficient to meet high standards of consent.

## Social Engineering

Privacy affects the present and shows signs of impacting our futures too, as new functions change what we understand by it. From this we can see that connecting up devices and creating new infrastructures has the potential to alter society at a more global level. It is capable of changing the culture around us. This change has a long-term trajectory and it is even more nebulous to consider in designing than privacy issues. Nonetheless, we raise the matter here, in the context of responsible design, because it gives something to think about in constructing the new world of connected things. There are likely to be unintended consequences.

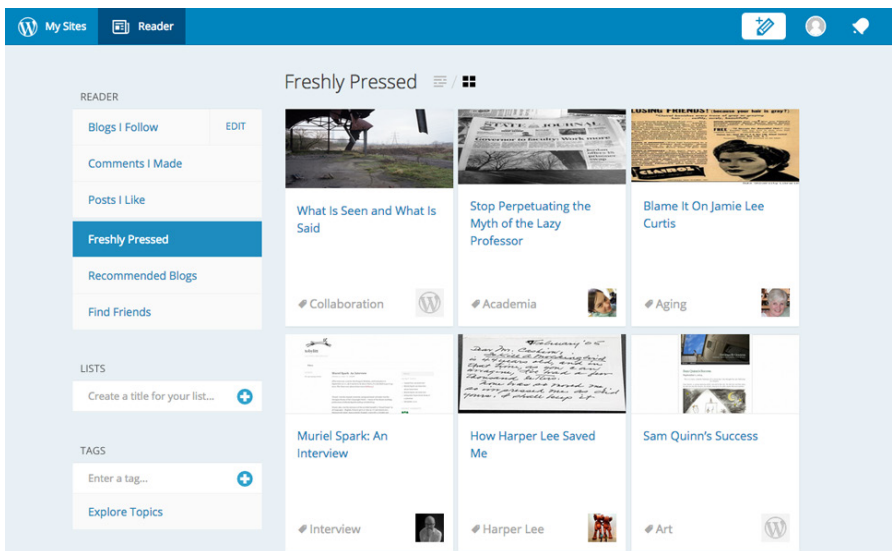
This is because we can never know the full range of impacts possible when we introduce the means to alter relationships in society. And the Internet of Things is predicated on altering relations. After all, things will be communicating their whereabouts, state, and ability to support



our needs. And we will be controlling them, displaced in space and time. We may not intend to be changing behavior with our designs, but over the longer term and the wider context, change is going to happen—related to how technology has been implemented.

To give an example, let's look at the implications of the personalized gathering and filtering of information that IoT enables.

In the early days of the Web, there was considerable concern about the “Daily Me” (a term we owe to MIT Media Lab founder Nicholas Negroponte; see Figure 11-16). This was a customized online news service that showed to you only what you were interested in and it was criticized for being too individualistic. More recently, *bubbling* in search results and social media has fueled the argument that we are fragmenting our cultures and losing access to the common experience that, among other things, encourages empathy and compassion. Bubbling is the tendency to show you what you want to see out of all possible feeds, based on previous choices, location, and other contextual factors. It builds “echo chambers” and keeps people within them. It is used to produce relevant results, but is accused of exploiting the human desire for confirmatory bias and reinforcing existing perspectives.



**FIGURE 11-16**

Zite.com (recently acquired by Flipboard) is a news aggregator based around preferences, similar to the concept of the Daily Me

With IoT, this filtering is no longer limited to online contexts. Now there are vending machines in Japan that will not sell prohibited goods to minors because they can read networked data. There are nearly six million vending machines in Japan, worth almost US\$60 billion a year in trade (see Figure 11-17).



**FIGURE 11-17**

A “smart” connected vending machine in Japan (image: Cameron Stone/camknows via Flickr)

Further, some machines use sensors to estimate the age, gender, and “type” of the person approaching the machine<sup>28</sup> and make recommendations accordingly. “For instance, for a young girl, the machine might recommend a sugar-free drink, while for an athletic young male it might recommend water or an energy drink, all based on market research and information collected from the machines themselves. The vending machines will also recommend drinks based on the time of day, season and temperature.”<sup>29</sup>

Extrapolate this and the connected world that a young woman experiences is markedly different from that of a young man. This would be true even if we only draw on the current marketing trends in play in

<sup>28</sup> <http://bit.ly/1In5sBj>

<sup>29</sup> Ibid.

the world, used to inform these machines' behavior now. But, this is not a static situation. The difference in experience serves to emphasize different patterns of taste and so the divergence, which was once subtle, can subsequently become quite extreme. Girls may never even see that beer is for sale. Boys will only be offered the chance to do “things that men do,” whatever version of masculinity has become enshrined in the information provided to the system. Now imagine that this is extended to one's identity as an ethnic minority too. If you are recognizably of a discernible minority, you will be presented with choices that reflect existing tastes and trends in that community. It is easy to see how this will encourage segregation rather than assimilation, divergence rather than equality.

Similarly, simple choices may matter for years to come. If you buy a health drink, you access a profile of yourself as concerned about fitness, so you see healthier options—and you live a healthier life. What you are offered by your smart environment takes you down one path, rather than another.

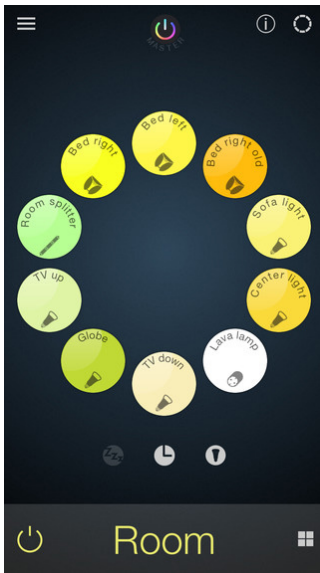
We are not advocating any particular way of living, but rather raising questions as to how we refrain from inadvertently introducing infrastructure so specific to how we live now that we prevent future societies pursuing a different track. Connecting up products produces particular relations, which are then potentially amplified, using correlational data, to offer normative choices back to users, in the hope of pleasing them. Clearly, this penalizes outliers and rebels. But it also results in a shift toward articulated norms for the majority. These norms will be much the same as those that are underpinning advertising already, but they will be imported into the fabric of homes and other new and intimate contexts. They will become linked to *who we are* as well as what we do. This tendency has been discussed in the context of the Semantic Web, with recommendations as to how to avoid closing down options.<sup>30</sup>

If we take this concern about reinforcing stereotypes back into the connected home, we see other potential points of tension. None of these are things that designers can legislate upon; all are things that could be considered in designing for connecting up products. For instance, what

---

30 A. Light, “HCI as Heterodoxy: Technologies of Identity and the Queering of Interaction with Computers,” *Interacting with Computers*, 23:5 (September 2011).

are the politics of the permission hierarchies in place to run the smart home? Who says what should happen and who can override others? How will this differ from the democracy of the visible switch, which anyone can flick and thereby undo someone else's choice? We need only look at the history of control of the TV remote to see that new systems disturb this democracy (see Figure 11-18). When the whole of life is administered by remote control panels that only some people can override (and may be invisible to others sharing the space), the politics of permission will be a live issue in many households, just as control of the TV was before people got their own sets.



**FIGURE 11-18**

A new generation of remote controls will introduce new power struggles into the household (image: iConnect Hue)

There is always the danger that such mechanisms impact most on the least powerful and most vulnerable. A particular field that is already contentious is care for the elderly. At its best, IoT technology can be a great enabler. At its worst, IoT technology can support the imprisoning of older people in safe, dull environments, without the need of much social intervention to manage their everyday functions. Sensory information about the neighborhood is reduced through environmentally controlled boundaries, and if the windows are not automated to reflect the weather, then a button reduces the effort of opening them,

thereby removing need for the movements that keep muscles ticking over. Because safer options exist, when residents choose risk, excitement, or undue physical exercise, they become a cause of concern for their care givers, in case they expose them to litigation.

And abuse can take other forms. While the new car clubs employ IoT technology to make hyperlocal use of shared resources possible, the same remote management tools are being used to penalize people with auto loans who are deemed a risk by their lenders. *The New York Times*<sup>31</sup> recently ran a story about people who had their cars repossessed remotely by lenders, who were able to interrupt the starter on the car if they were concerned about late payment. This disconnection happened sometimes while the client was driving—there is one story of a woman caught without power in the middle of a major road (see Figure 11-19). None of these drivers had yet defaulted on payment; they were merely a few days late. Such an aggressive protection of assets is not restricted to cars, but cars are particularly an issue, both because the crude use of remote disconnection could potentially be fatal, and because so often the car is a lifeline for families with vulnerable dependents. Manufacturers argue that tools such as this mitigate risk and make it viable for lenders to offer loans to consumers who might otherwise not qualify for credit at all. But disconnection without warning is far swifter than current forms of retrieval, and leads to subprime borrowers being penalized more readily for minor late payments than their better-off counterparts. Vehicles have GPS tracking fitted too, thus both safety and privacy are potentially compromised.

A final situation we can consider when we think about social engineering might sound futuristic, but it is being discussed as autonomous systems start to find a niche in the market. There is already speculation about what should happen ethically and legally when a networked machine is acting on a person's instructions but using its own "initiative" in choosing how to carry them out.

---

31 M. Corkery and J. Silver-Greenberg "Miss a Payment? Good Luck Moving That Car," *New York Times*, September 24, 2014, <http://dealbook.nytimes.com/2014/09/24/miss-a-payment-good-luck-moving-that-car/>.



**FIGURE 11-19**

A GPS tracking unit with the ability to disable a car's starter motor (image by thematthatter on Instructables.com, shared under a CC license)

You may want to consider driverless vehicles on roads to grasp the import of this, as they overcome the problems of interacting with other cars, pedestrians, and cyclists. But, before finishing this section, we might also contemplate briefly what the next wave of systems brings with it. The field of artificial intelligence has been working for years to create systems that adapt their behavior to new situations by “learning” about them. These systems are not just interacting with a set of conditions by choosing a good path of action in response, they are adapting to the conditions they find repeatedly by altering their potential for future behavior based on their experiences—as people do. Ethicists have already begun discussion about how responsibility for this unspecified behavior can be assigned, whether it is reasonable to hold designers responsible for the actions of a system that could not be debugged in every possible situation, and what the ethical, social, and legal implications of coexisting with learning networks might be. Autonomous systems, especially those with the power to learn, are theoretically unpredictable, even if they never actually do anything controversial at all.

Should current trends continue, there will be an added layer of complexity to designing people's relationship with technology that hangs on how far and how safely the technical tools can adapt.

### **Mitigating the dangers of social engineering?**

One way of avoiding the worst of these scenarios is to use methods that mean you design *with* your users as well as for them, listening to them and mitigating their concerns (see Chapter 6). But even participatory design<sup>32</sup> will not tackle all the bigger trends that are subterraneously shaping choices of how technologies are implemented. Some aspects of these relations will only be held up to the light as new forms of interaction make them visible. That argues for systems that can be reconfigured easily as we all learn about what they enable.

So it also behooves those with responsibility for design to take their power seriously and design systems that are easy to use, easy to escape from, and do not lock people in, either literally through malfunction, or figuratively, through a desire for short-term gain. And as more automation and machine learning is incorporated, there is also a need to acknowledge that it is not always possible to predict what a system will do as it grows and learns.

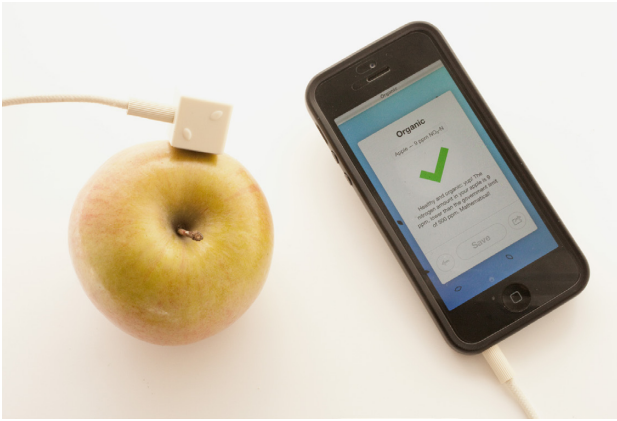
In summary, the long-term trajectory for connected products is one that impacts on the local control of ordinary people, and eventually, on identity and self-determination. Yet, no dystopia is assured, any more than a utopian new world can be forecast. What we can predict is that there will never be a dull moment for anyone interested in the ethical aspects of implementing these technologies.

## **Environment**

On first impression, there seems to be a theme of environmental awareness running through the current crop of consumer IoT systems. Smart meters will help us understand our energy impact, and devices such as smart thermostats will help us waste less of it. Sensors monitoring air and water quality and radiation can help us better understand our impact on the planet (see Figure 11-20).

---

32 The Proceedings of the 1990 Participatory Design Conference.



**FIGURE 11-20**

The Lapka organic matter sensor detects levels of nitrates in fruit and vegetables (image: Alex Washburn/CC license)

But look a little deeper, and there are valid concerns. The tech industry scramble to turn out the next big product used to mean ones and zeros, but now more hardware is being manufactured. As Allison Arieff points out, the maker movement is enthused by the potential of home 3D printing, but fails to consider that it uses 50–100 times more energy than injection molding and increases the use of plastics.<sup>33</sup>

And in the past, our nonconnected devices were thrown away only when they stopped working and were uneconomic to repair. But the lifecycle of computing devices is far shorter. Devices that still technically work become obsolete much more quickly. Will IoT result in us making, buying, and throwing away more stuff?

This need not always be the case. Connectivity can recast products as shared services, such as reducing the need for car ownership among occasional drivers through car clubs (see Figure 11-21). It can enable better use of the resources we have, such as through smart car parking schemes that reduce the amount of fuel drivers burn (and pollution they create) driving around looking for spaces.

---

<sup>33</sup> <https://medium.com/re-form/just-because-you-can-doesnt-mean-you-should-252fdbcf76c8>





**FIGURE 11-21**

Car clubs, such as Zipcar, have been shown to reduce car ownership (image: Zipcar)<sup>34</sup>

But it's hard to understand what the true positive or negative environmental impact may be of the systems we create. In this section, we look at factors that may affect the impact of IoT systems, and what designers' responsibilities may be.

## **WHAT'S THE ENVIRONMENTAL IMPACT OF AN IOT PRODUCT?**

Ethically, we should ask ourselves how our systems may be doing both good and harm. Standards and certification bodies provide limited information and legal controls on some aspects of products. But assessing this accurately requires us to look at the environmental impact of a product over its entire lifetime. For that, no complete picture is available.

### **Manufacturing**

The first impact of manufacturing is in resources. Electronic devices tend to contain more precious and rare earth metals (e.g., copper, gold, palladium, and platinum). These are valuable resources yet often lost when old products are not recycled (only 1% of cellphones in the United

---

<sup>34</sup> [http://www.uctc.net/access/38/access38\\_carsharing\\_ownership.shtml](http://www.uctc.net/access/38/access38_carsharing_ownership.shtml)

States are recycled<sup>35</sup>). Mining them consumes a considerable amount of energy and has environmental consequences. For example, radioactive waste (thorium) is a by-product of rare earth metal mining. In some cases, such as tantalum (mined from coltan and used in capacitors), ethical questions have been raised about the corporate responsibility of the mining companies. Tantalum is even classified as a conflict resource due to links between coltan smuggling and warfare in the Democratic Republic of Congo.

Electronics also contain materials that are toxic to humans and the environment, such as lead and cadmium in printed circuit boards, brominated flame retardants used in PCBs and plastic casings, and PVC coatings on cables and casings which release dioxins and furans when burned. Any product sold or imported into the European Union must comply with the Restriction of Hazardous Substances Directive (RoHs), which restricts the use of toxins such as lead, mercury, and cadmium. But even when individual devices contain lower levels of toxins, a large quantity of discarded devices can still be highly toxic.

Manufacturing devices also consumes energy and creates greenhouse gases (this is described as the device's "embodied energy/carbon"). Nest addresses this up front on their blog, claiming that it takes 8 weeks for their thermostat to become carbon neutral by offsetting the energy involved in manufacturing and distribution.<sup>36</sup> But devices that do not reduce the user's overall energy consumption cannot claim the same virtuous effect.

## Usage

During usage, the device should make efficient use of energy (for many embedded devices that run on batteries, this is not just an environmental concern but essential to the normal functioning of the system). Energy Star, a program of the US Environmental Protection Agency, sets globally used standards for energy efficiency in consumer product specifications.

---

35 <http://www.pbs.org/wgbh/nova/next/physics/rare-earth-elements-in-cell-phones/>

36 <https://nest.com/blog/2014/04/22/todays-earth-day-tomorrow-should-be-too/>, via <http://www.wired.com/2014/06/green-iot/>

The total energy consumed by the product also includes the energy required to provide the Internet service. In 2010, data center electricity consumption accounted for an estimated 1.3% of all electricity use in the world, and 2% of all electricity use in the United States.<sup>37</sup> Google themselves claim that doing a search consumes 0.0003kWh of energy,<sup>38</sup> the equivalent of running a 60w lightbulb for 17 seconds. That might not seem like much, but across billions of potential users and devices, the energy consumption of an Internet service would be huge.

### **Maintenance and upgradeability**

In theory, providing a device with Internet connectivity ought to extend its lifespan. It should be relatively easy to deliver software or firmware updates that improve the device's functionality and performance and keep it in service for longer. You could even charge for software updates in lieu of new hardware. However, in practice, increases in functionality tend to make ever-increasing demands on hardware, which is usually manufactured to a tight budget and may not have the additional processing power, memory, or UI capabilities to handle very many future upgrades. At some point, the old devices may no longer be powerful enough to support the latest and greatest software. It can also become uneconomic for manufacturers or service providers to continue to support software updates and security patches to a large number of older versions of devices out in the field (as we saw earlier in the "Security" section).

A nonconnected home lighting or heating system might be expected to last 10–20 years (or more, in the case of lighting), but its connected equivalent is likely to become obsolete much faster. Even if it doesn't fail, it may no longer offer us the new functionality we have come to expect from such systems. And products are explicitly designed for planned obsolescence: companies want us to buy new things.

### **Disposal**

Finally, there is disposal: retiring devices at the end of their lives. This creates waste, some of which is highly toxic. Electronic waste represents 2% of America's waste in landfills, but is responsible for 70% of

---

37 <http://www.analyticspress.com/datacenters.html>

38 [http://www.theregister.co.uk/2013/08/16/it\\_electricity\\_use\\_worse\\_than\\_you\\_thought/](http://www.theregister.co.uk/2013/08/16/it_electricity_use_worse_than_you_thought/)

the heavy metals in that waste.<sup>39</sup> Chemicals from landfill leach into the groundwater. Very few PCs and smartphones are currently recycled. According to the US Environmental Protection Agency, fewer than 10% of the 141 million mobile phones discarded in the United States in 2009 were recycled.<sup>40</sup> In the European Union, the Waste Electrical and Electronic Equipment Directive (WEEE) mandates that it is the responsibility of the producer or distributor to make provision for the collection and recycling of electronic equipment placed on the market after 2005 (see Figure 11-22).



**FIGURE 11-22**

E-waste waiting for recycling at a specialist processing facility in Kent, UK (image: SWEEP Kuusakoski)

Recycling helps recover valuable materials such as metals, but it's not a perfect solution. In *Cradle to Cradle*, William McDonough and Michael Braungart argue that recycling is generally downcycling: converting materials into lower-quality materials with a more limited range of uses.<sup>41</sup> The universal recycling symbol (Figure 11-23) may be a loop, but this is misleading. We should really be designing materials that can be recycled into equally high-quality materials.

---

<sup>39</sup> [http://www.cleanair.org/waste\\_delete](http://www.cleanair.org/waste_delete)

<sup>40</sup> [http://rmcp.earthworksaction.org/why\\_recycle#.VFkWTSSsU4J](http://rmcp.earthworksaction.org/why_recycle#.VFkWTSSsU4J)

<sup>41</sup> William McDonough and Michael Braungart, *Cradle to Cradle: Remaking the Way We Make Things* (New York: North Point Press, 2002).



**FIGURE 11-23**

The universal recycling symbol implies that materials can be recycled into equivalent materials, which is not the case (image: C Buckley via WikiCommons)

### **WHAT CAN DESIGNERS DO?**

IoT products may offer the promise of reduced energy consumption, better understanding of the world, and more efficient use of resources. But if we approach creating connected products the way we've approached everything else, we're at risk of accelerating the destruction of our environment through burning resources making disposable junk. We need to reduce the impact of the systems we create and not add to the damage.

We can say “no” to making pointless stuff that will end up in landfill in six months. We can design things that reduce consumption by reducing waste, or supporting usage, not just ownership. We can consider the materials we use, and how things are made and design for end of life and the next life of materials, not just this one. We can choose responsible hosting companies who try to conserve energy and invest in renewable power. We can design for upgradeability and longevity through software updates (see Figure 11-24).

That may sound idealistic, but it's not intended to deny the need for most products to make money, or our desire to have good products in our lives. William McDonough and Michael Braungart propose a triple bottom line approach to business strategy based on economy (making money), equity (providing value and treating people fairly), and ecology (not harming the environment). The intention is to help the world prosper through consuming. Taking nature as inspiration, a production process in which everything is used and returned to the cycle can do good instead of harm. But there are no easy answers.



**FIGURE 11-24**

The Fairphone smartphone uses conflict-free minerals and is designed for repairability and upgradeability<sup>42</sup> (image: Fairphone)

## Summary

Embedded devices are especially vulnerable to security threats such as eavesdropping or malware. They may not have been designed with security in mind, and keeping security software up to date, if they have any at all, may be difficult or impossible. They may also lack ways of indicating when they have been compromised. Security is often complex to use, and in tension with usability, as it creates barriers to the user instead of making things easier.

Usable security for IoT is a major challenge, and will need to focus on limiting the damage that can be done when a system is compromised as well as making it easier for users and devices to authenticate themselves and for users to set permissions controlling what devices and applications can do.

Privacy is a many-headed problem in smart contexts. Systems may reveal information about others' behavior to people who share the same space. Providing a service requires that potentially compromising personal information is gathered and aggregated, leaving people open to privacy violations from service providers and third parties. And data

---

<sup>42</sup> <http://www.fairphone.com/>

can be combined and mined for details that are not obvious to those providing it. To cooperate with data protection legislation and protect users, *Privacy by Design* is a means to take the minimum data needed, work with it in anonymous form, and delete it as soon as practicable.

Meanwhile, systems that treat us differently according to the details they hold about us can produce unanticipated social consequences by amplifying differences along stereotypical lines. The new services may contribute to unintended social engineering. The best way to resist this is to work with users to identify likely problems and build in flexibility at outset, but effects may be hard to predict ahead of implementation.

Energy and environmental monitoring are strong themes in IoT. But there are real risks that the technology rush will have a damaging environmental impact through the consumption of resources and creation of more e-waste. Designers need to consider the environmental impact of a product across its entire lifecycle, taking into account manufacturing and materials, energy consumption during usage, designing for maintenance and upgradeability, and ensuring high-quality materials can be recovered through recycling at end of life.





# Supporting Key Interactions

**BY CLAIRE ROWLAND WITH MARTIN CHARLIER<sup>1</sup>**

This chapter sets out some practical requirements for key interactions that are common to different types of connected products.

There will be many aspects of your design that are not covered here as they will be unique to your product or market. But regardless of what your product does it will almost certainly need ways to do some of the following:

- Get the system set up and devices connected
- Access controls and data
- Manage devices and alerting behavior
- Configure and manage automated behaviors

In some interactions, such as device pairing, technical requirements will mean that your design has to follow a tightly scripted flow. We've described these in detail. In other interactions, such as accessing controls, there may be many possible design solutions. Here, we've shown examples of common design approaches currently used in live products to provide starting points for solutions.

We would not want to limit your creativity by suggesting that any current design reflects the “best” way of doing things. The field of IoT is far from ready for any defining design patterns since we have so many suboptimal situations in both hardware and design, as the rest of this book discusses.

---

<sup>1</sup> Thanks to Gawain Edwards, Alex von Feldmann, Anders Mellbratt, Niel Bornstein, Liz Goodman, and Mike Kuniavsky.

However, requirements and design problems across various types of system in the market right now have similarities. And we can draw from existing approaches in the interaction design arena to increase our confidence in drawing up these issues. As with much of this book, the examples here come heavily from connected homes with some from different contexts, such as wearables. Not all of these will be relevant to all types of system, but there are commonalities across different systems.

Use this chapter as a checklist of possible requirements and source of examples. We hope it will save you some effort specifying and designing the necessary functionality to get and keep your system up and running.

This chapter introduces:

- Practical design requirements for getting up and running with the system, connecting devices and user accounts (see page 469)
- Typical design requirements for in-life “housekeeping” functionality: the unglamorous but important admin functions your system will need in order for users to get and keep it up and running, such as managing devices, handling notifications, user management, and system maintenances (see page 479)
- Emerging design approaches for system discovery (see page 486) and control (see page 487), experiences that are common to many systems on the market today
- Designing for platforms: if your system supports multiple applications, you’ll need to consider common design components to make your UX feel coherent (see page 494)
- In-depth technical information on connecting up and pairing devices (see page 499)

This chapter addresses the following issues:

- Why flexible, modular setup processes help users on first time and subsequent use (see page 471)
- Why you need to consider the flow of setup instructions across print, UI, and packaging (see page 474)
- Why users need a way to manage devices, but showing signal strength is misleading (see page 480)

- Why dashboard-type designs for controls don't scale (see page 488)
- Why your platform needs a consistent design language (see page 497)
- How to get devices with and without screens and keyboards connected to the network (see page 499)
- How pairing works, and the different patterns Bluetooth supports (see page 509)

## Setup

Ease of installation is vital to a good experience, especially if consumers are expected to complete it themselves. A good setup experience by itself is not the only thing needed to make a service successful, but a bad setup experience can be enough to make it fail. If the setup experience of a product or service is bad, the customer may lose enthusiasm for the product or give up on it altogether. They may also install it incorrectly, so it never functions quite as it should. For example, a well-known brand of video baby monitor received poor reviews when even engineers could not figure out how to connect it to WiFi. Returns of devices such as broadband routers and cellphones where no fault is found with the equipment are estimated to cost the global electronics industry billions of dollars per year.<sup>2</sup> A significant proportion of these are likely just to have been misconfigured, often due to the poor usability of the setup process. We can probably expect IoT systems to follow similar patterns.

Setup is hard to cover in general terms, as it varies so much from product to product. The process of installing devices in a home or other location is very dependent on the device and what it does (and of course the characteristics of the user's home and their personal goals for using the product). Some devices, particularly those being wired into the fabric of a building, may require professional installation.

---

<sup>2</sup> "‘No Fault Found’ returns cost the mobile industry \$4.5 billion per year," WDS, <http://www.wds.co/no-fault-found-returns-cost-the-mobile-industry-4-5-billion-per-year/>, and "Reducing Product Returns: U.S. Consumers Want More Product Information from Manufacturers & Retailers," *Appliance Magazine*, October 2002: <http://ubm.io/t15Ly0W>.

One of the most critical activities is getting devices connected to each other and/or the Internet. Once devices are connected, there will often be some configuration and customization to do to get them working the way you want. Services involving multiple devices and automated behaviors or rules, such as home monitoring kits, can require quite intensive configuration.

A good setup UX is important. But what happens after setup—the in-life experience—needs careful design too. When you’re figuring out how your system should work, there may be certain technical decisions you make to enable easier setup. (Adding support for out of band pairing, described in the “Getting Things Connected” section, is one example.) But you also need to have a very good idea of what the in-life experience will be. Designing an effective setup experience requires the user to understand what they are setting up and why, and you can’t do that unless you know that too.

In this section, we’ll look at overall patterns for helping users with setup.

## **CREATING GOOD SETUP EXPERIENCES**

Setup is hard to simplify. It often forces users to engage with some activities or parts of the system that they don’t understand very well. Key questions to ask when you’re designing a setup experience are:

- Are there elements of the system we can supply preconfigured, or must the user set everything up? What defaults, if any, will create problems if not changed? (For example, not changing the default admin username and password on a router will leave it open to malicious use.)
- How much flexibility can we allow the user in terms of the setup process? Must everything be done in a very precise sequence, or can the user be allowed to complete the steps in any order?
- What are the best ways to provide instructions and feedback?
- Is anyone else involved in setting the system up?
- What’s the end point of setup? How can you support the user to a point where they are getting real value from the system?

- How easy is it for the user to make changes after initial setup and configuration? Users may need to duck back into parts of setup later on to add or remove devices, reconfigure functionality, or even pass on ownership of the system.

## **PRECONFIGURATION VERSUS USER EFFORT**

The ideal user experience is as little setup as possible (i.e., “plug and play”). Devices might come with batteries preinstalled, or be already paired to the hub. But the additional effort and cost for the supplier may make this unrealistic. And the more preconfiguration you do, the more you have to make assumptions about what the user wants... which may be wrong.

Pilgrim Beart, in his role as founder of AlertMe, a UK-based connected home technology company, talks about the challenges of designing the setup of the company’s original consumer home monitoring kits (see Figure 12-1):

“Ideally you’d open up the box and it’s configured for you, the sensors would have the right names, but this adds a lot of complexity to the supply chain. So you have to get the customer to do the customization. One variation could be that you could preconfigure the kit for the most typical home, and then the user would customize it. But there’s no such thing as the typical home, and people can get upset when you imply they are not typical or make assumptions.”

## **FLEXIBILITY AND MODULARITY**

Most systems take the procedural approach of walking users step by step through a process. In theory, they can succeed in setting up the system without needing fully to understand what’s happening. This approach can be slow for more knowledgeable users but for a one-time interaction for inexperienced users, clarity is usually more important than speed.

The chief drawback of a rigid process is the risk of the user running into difficulties if they do something out of sequence, either by mistake or by accident. Users often don’t sit down and work through your instructions step by step: they often expect to be able to muddle through without reading them at all. If they have done things outside

your predetermined sequence, the system may now be in a state that does not match anything in the instructions. They may be unable to resume where they left off.



**FIGURE 12-1**

The original AlertMe home monitoring kit

It's better, if possible, to provide a degree of flexibility. Certain sequences of actions, such as connecting up a gateway, might have to happen in a certain order. But it probably doesn't matter whether the user pairs all their devices before seeing the app homescreen. If you can detect what the user has already done and still needs to do, you can allow them to complete those steps in whatever order (see Figure 12-2). This supports the high proportion of users who will pile into setting up the system without bothering to read the instructions... until they get stuck.

As Pilgrim Beart says:

Forcing the user to set up their new system in one particular “logical” order just doesn't work. We originally thought there was an “obvious” order: plug in the hub, register it with the cloud, put batteries into devices, name the devices. So that's what we designed for, and put in the instructions. But it turned out that this was just not intuitive to many users, many of whom didn't read the instructions either, which led to them getting stuck and needing customer support, which

doesn't scale. So to make setup truly a no-brainer we had to put in a lot more thought and engineering work, to allow users to do things in whatever order makes sense to them, with the system reacting to give them appropriate guidance on how to complete the process. This puts the user in charge, with the system as helper, not the other way around.



**FIGURE 12-2**

The Lowes setup process (based on the AlertMe platform) allows users to break off and resume setup later (image: Lowes)

Where possible design for interruptions: allow users to complete part of the process, pause, and return to it later. If users have to complete the entire process in one sitting, they must have all the necessary information and materials on hand, and find a time when they can safely ignore the doorbell, incoming phone calls, and any other interruptions. With the best of intentions, this is often impractical. They will also need to have a good understanding of what they are doing, as making (and undoing) mistakes or stopping to refer to the instruction manual or contact customer support will break the flow. This could further increase confusion, and thus the risk of getting into more serious trouble. Stuck in a rigid, poorly designed process to set up a broadband router, one of the authors of this book managed to lock herself out of all commands including the factory reset. She had to buy a new router.

One connected home system we encountered required users to complete setup in a single sitting, taking up to an hour. If you only managed part of it, it wasn't possible to resume where you left off. If you had set up your user account before pairing all the devices and were then interrupted, the system would log you out. Logging back in would take you to an empty homescreen, showing no devices, without any status information or obvious action to add them. The system just looked broken. A better experience would be to show the homescreen with a prominent link saying, for example, "have you added all your devices?"

However you choose to walk users through setup, ensure it's always clear exactly where they are in the process, what's just happened, and what will happen next.

It can help to set some upfront expectations about what will happen at the start of setup. Having an understanding of the overall process and objectives will make users feel more confident and help them orient themselves better if things go wrong. This plays to different cognitive approaches that users may take. Some may need a map to orientate themselves; others (especially novices) will still need step-by-step instructions.

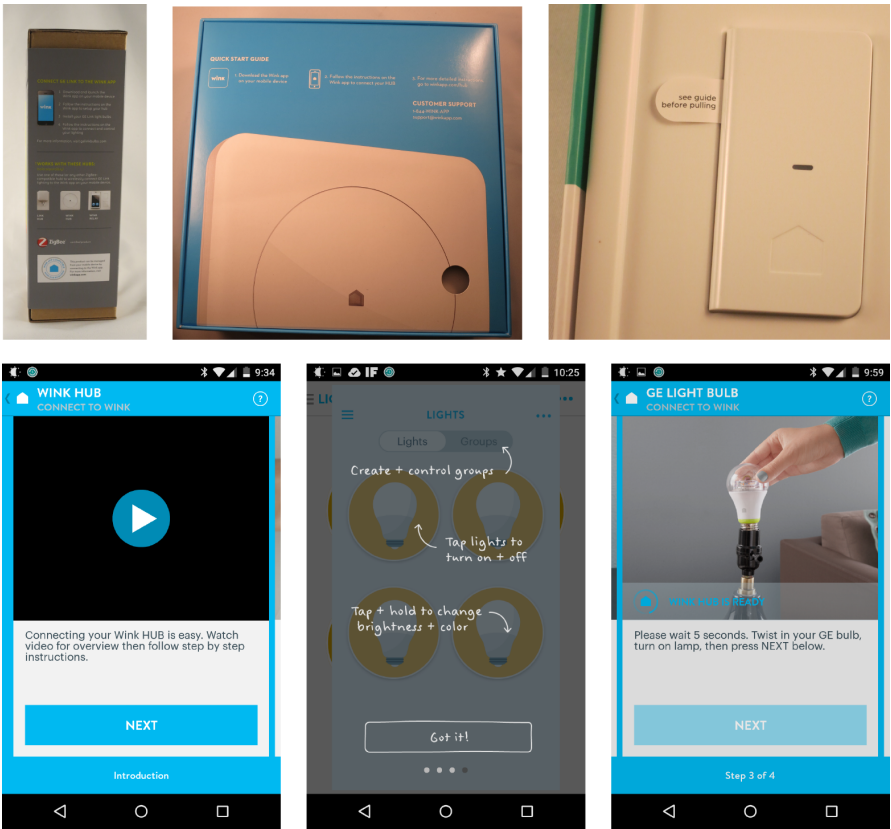
Some of the functions that are part of setup may be things the user will need to go back and do again in the future. Examples include replacing batteries, adding or removing devices, and running software updates; even changing owner, linking to a new account, or moving house. A flexible and modular approach to designing the setup experience, that allows the user to dip in and out if necessary, may help you design those standalone functions more easily.

## **INSTRUCTIONS**

Many systems provide instructions via a combination of media, such as printed instructions, videos, in-app (web or mobile) guidance, and physical labels on devices (see Figure 12-3). Each of these has valuable uses. For example, a video can be a great way to show complex actions like removing a battery cover, or sequences of interactions, like just how an LED will flash during pairing. Color-coded physical labels can be a nearly foolproof way to indicate which cable needs to be plugged into which port. Contextual instructions that provide onscreen guidance as part of the setup process are ideal. For example, the guidance



that tells you how to put the battery in the motion sensor should be available on the same screen that indicates whether the motion sensor has been detected by the system.



**FIGURE 12-3**

Product instructions, like those for Wink, can be spread across packaging, apps, and any connected devices that have screens

But the fact that instructions are distributed across multiple places can also add complexity. You need to take a holistic, service design approach to designing setup experiences. Consider the sequence of actions required; the packaging, devices, UIs, and instructional media that may be used at each stage; the handoffs or transitions between them; and the context of use at each point. (The BRCK case study in between Chapters 12 and 13 describes how the designers went about designing the setup UX, spanning hardware, software, and packaging/instructions.)

If the same instructions are available in more than one format, for example, print and web, ensure that *either* both sets are complete and clearly carry the same content, *or* that each has a clearly defined and different role. This means it's clear which part is handled by which medium. For example, the print guide helps you get the device powered up and the control app installed, from which point onward you see instructions in the app itself. Users should not have to waste time comparing print and onscreen guidance and worrying about which they should be following.

If you use partial instructions in different formats, ensure there is clear signposting across media. For example, a printed quickstart guide in the product box might give the user just enough information to get the product connected (see Figure 12-4). After that, they should be using a mobile/web interface or the embedded device interface itself if it has one. The switch between the two should be an explicit handover—for example, the print guide should say “now refer to the setup instructions in the mobile app.” Bear in mind that once users have a working system to interact with, their attention will be on that and they may stop looking at print instructions. Keep instructions close to their line of sight for performing the action if possible.



**FIGURE 12-4**

A cross-media handover in instructions on the Pebble

In particular, ensure wording and diagrams used on different media are 100% consistent with each other and the actual product. The print diagram used to illustrate how to put the battery into a contact sensor should look like the actual sensor, and be the same one you use in the UI. This can be a challenge, as it's much harder to make updates to print instructions than to onscreen ones. For example, a set of printed setup instructions was supplied with a home monitoring kit composed of security and safety sensors. The provider had commissioned artwork for the setup instructions at the time the kit was first released, but over time had switched to another manufacturer for one of the sensor components. The device was now subtly different to the older version of the diagram. However, they had not had time to update the artwork or the printed instructions. The printed setup instructions showed a diagram of the older version of the sensor, and while it looked similar, the method for opening the battery casing was slightly different. This caused confusion: it wasn't clear whether users had correctly identified the component, or why the casing wouldn't open as per the instructions. Were they doing something wrong or was it broken?

If there are lots of components, such as multiple motion sensors, or components are likely to be unfamiliar to users, consider putting physical labels on them so users can identify them out of the box.

UX designers Harry Brignull and Pete Gale's presentation "Out of Box Experience Design"<sup>3</sup> provides user research-based advice on the design of broadband router setup instructions. Much of the advice would also be applicable to a multidevice IoT system.

## CONTEXT OF USE

With any system that's designed to be used in a specific place, consider the context of use of setup. For example, in a connected home system, users may be required to connect all the sensors or devices to a hub or gateway, then physically place them around the home. Depending on the size of the home and range of the network, as they are placed around the home, some may lose connectivity to the hub. If the user has to walk around the home, installing devices, it would be useful to track whether or not they are still connected. If the device itself is too

---

3 <http://www.slideshare.net/harrybr/out-of-box-experience-design-presentation>

limited in capability to display this, this is an ideal opportunity to use a mobile device over a desktop. Even if setup is long and involved and some of it is better suited to a desktop computer, it's worth allowing users to switch input devices during installation.

### **IS ANYONE ELSE INVOLVED?**

For certain types of home system, such as heating, professional installers may be involved. You'll then have to decide which tasks are best done by the installer and which by the user. The installer may have greater technical knowledge but it may not be appropriate for them to complete certain steps themselves. Contractors may be short on time, and their employers may only allow them limited Internet access on their work devices. And the householder may feel more comfortable doing certain steps (such as setting their online account password) themselves.

### **GETTING THINGS CONNECTED**

The most critical stage of setup is getting everything connected. If there's a dedicated gateway, it will need to be connected to the Internet, over WiFi, Ethernet, or perhaps cellular data. It will need to be associated with a user account on the Internet service. For edge devices that go directly online, the steps are similar. There are different ways of approaching the design, depending on whether the device or gateway has a screen and input capabilities, or not. If edge devices connect via a gateway or to each other, they will need to be paired, perhaps over ZigBee, ZWave, or Bluetooth.

It's useful to understand some of the technical requirements for getting things online and pairing, as these will determine much of the interaction flow. We've discussed the technical requirements in detail toward the end of this chapter, in "The Technology of Getting Things Connected."

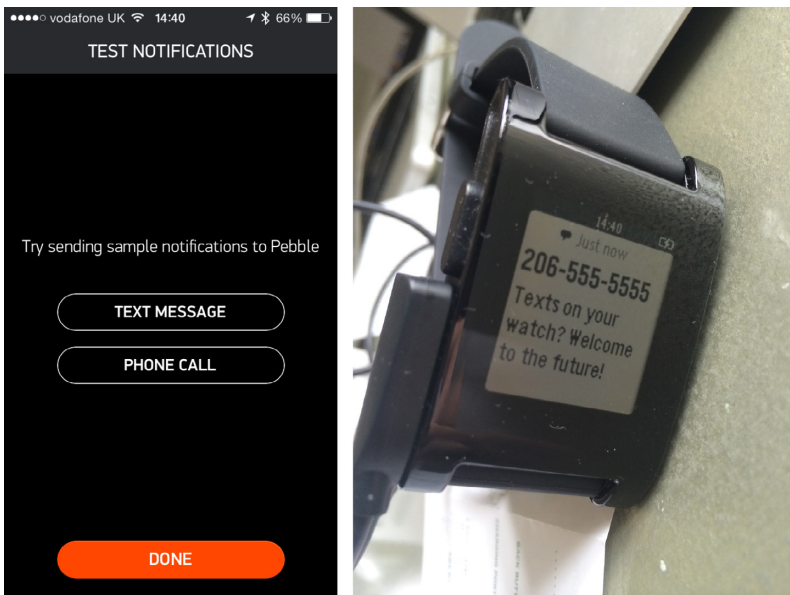
### **GETTING REAL BENEFIT OUT OF THE SYSTEM**

Setup doesn't end when the user has physically connected all the devices. Quite often, extra configuration is needed before the system can deliver real benefit to the user. For example, once a user has connected up all their security sensors, they will need to identify the most likely entry points for an intruder and place the sensors around the house. Or, having set up a lighting system, they may at some point

want to configure groups and scenes of lights. Users are likely to need guidance with this, but don't overwhelm them with too many options all at once. Get them to configure the basics first, then link to more guidance as they are ready. Sometimes it may be better to let them get used to using the basics for a while, then make suggestions for getting more out of it. Think of this not as presenting everything at once, but designing a journey through which users learn more about the system, and it becomes better adapted to their needs.

## TEST IT'S WORKING

Once everything's connected, you may want to include a test mode to verify everything is working. That might include testing connections, controls, or notifications (see Figure 12-5).



**FIGURE 12-5**

Pebble offers an option to test notifications

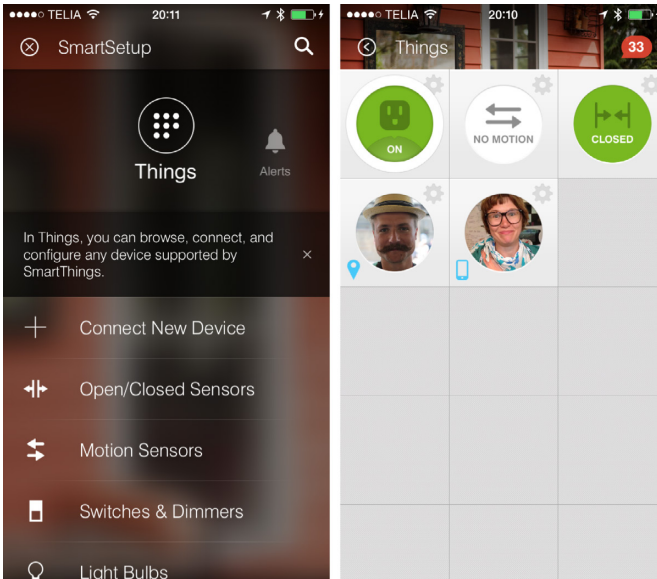
## In-Life Housekeeping

Once the system is up and running, there are some other basic housekeeping functions that users are likely to need.

## DEVICE MANAGEMENT

Users are likely to need ways to add more devices, remove ones they no longer want, and manage the settings of devices. If there are multiple devices to manage, then a central location for viewing and managing all of them makes sense (even if this functionality can be accessed from multiple places in the system).

A typical approach is to have a list of all the devices on the system showing some basic information (see Figures 12-6 and 12-7), with more detail available on dedicated screens for each device (see Figures 12-8 and 12-9). What information you choose to show and where will depend on the type of system you're creating. But common elements include the device type, name, whether it's connected OK or not, any current status information (e.g., on/off or last sensor data reading), when it last checked in/sent data, battery status, any metadata you have added (like its location), and, if appropriate, any functionality with which it is associated (such as automated rules). There might also be the option to identify it, through a beep or LED flash (this is useful if there are multiples of that type of device and you need to know which is which).



**FIGURE 12-6**

The Things list in SmartThings



home | devices | contacts | magic | history | account | store | help | sign out

devices Add Devices

Device	Health	Power	
Home Iris Smart Hub	All ok		<a href="#">Manage</a>
Contact Sensor 1 Contact Sensor	All ok		<a href="#">Manage</a>
Contact Sensor 2 Contact Sensor	All ok		<a href="#">Manage</a>
Keypad 1 Keypad	All ok		<a href="#">Manage</a>
Motion Sensor 1 Motion Sensor	All ok		<a href="#">Manage</a>
Smart Plug 1 Smart Plug	All ok		<a href="#">Manage</a>

[Advanced Options](#)

© 2015 Lowe's. All rights reserved. Lowe's and the gable design are registered trademarks of LF, LLC.  
[Terms & Conditions and Privacy Notice](#) | [www.lowes.com/iris](http://www.lowes.com/iris)

**FIGURE 12-7**

The web UI device list from Lowes Iris

TELIA 3G 15:10

Anders 7

Beep

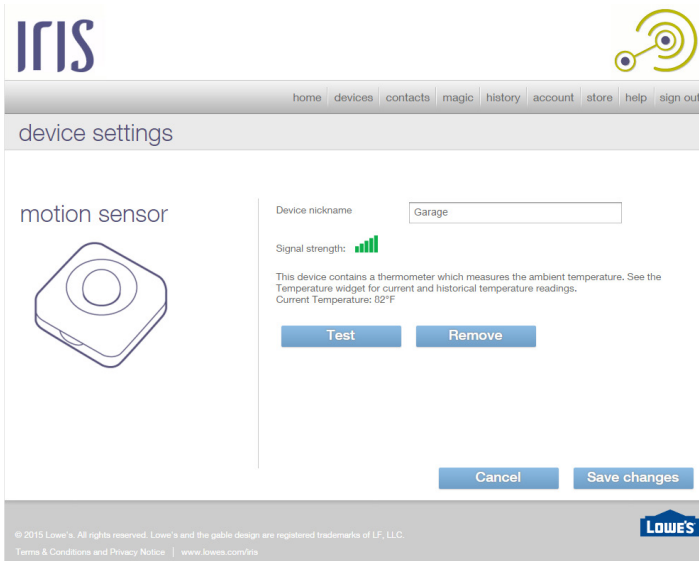
38% battery

SmartApps Shop Activity

Preferences

**FIGURE 12-8**

A device screen for a keyfob from SmartThings; note the option to make the fob beep, to identify or locate it



**FIGURE 12-9**

A device screen from the Lowe's Iris web UI

Think carefully about how much information users need, and what they can actually make use of. Independent IoT consultant Gawain Edwards offers a tip, learned from testing connected home systems: “Never, ever, show signal strengths to users. It unnecessarily worries people that they don’t have a perfect signal even if their devices work perfectly.”<sup>4</sup> Interference matters more than the strength of the signal: an edge device can often work very well with only a weak signal to the gateway if there is no radio interference from other sources. Just show the user that the device is connected, and warn them when it’s not working.

## **NOTIFICATIONS, MESSAGES, AND ALERTS**

If your system uses one or more types of notification to alert users to insights or state changes within the system, you may need to provide a place to manage and customize these. These might be forms of mobile notification (e.g., banners, badges, alerts), alerts on the edge device (like a smart watch’s onscreen notification or wearable vibration), or

---

4 In conversation.



using channels like email, SMS, or even triggering automated phone calls. In “Control Experiences,” we discuss activity feeds that keep the user informed of overall system functionality.

What you choose to communicate depends on the nature of your system, but some basic, system-related notifications might include low-battery warnings and devices being unavailable. There may be certain functionality that’s so central to your system that you need to let users know right away if it’s not working. For example, if a home security system or fire alarm is offline, it’s very important that the user is aware. The absence of an alarm usually means everything is OK, but if the alarm loses connectivity then no alarm is not necessarily good news. In other cases, a temporary outage might not be a problem. Home heating will continue to work even if Internet connectivity is lost. In that case, rather than notifying the user every time the Internet connection drops out, you might just tell them at the point where they try to use it. The key question is, would the user behave differently if they knew?

Choosing the correct communications channel is a question of how urgently the message needs to get through, how timely the information within it is, and how important it is that user takes action or not. You don’t want to find out your house is on fire via an app badge. And an alert to tell you that your home is currently using an unusually high amount of energy is probably not best delivered by email. If you are at home, you need to know right now, and the email may take time to get through. Consider also whether you need to provide feedback on follow up. For example, if your elderly grandfather has used his emergency alarm, it’s reassuring to know that your uncle has already responded and is on his way to help.

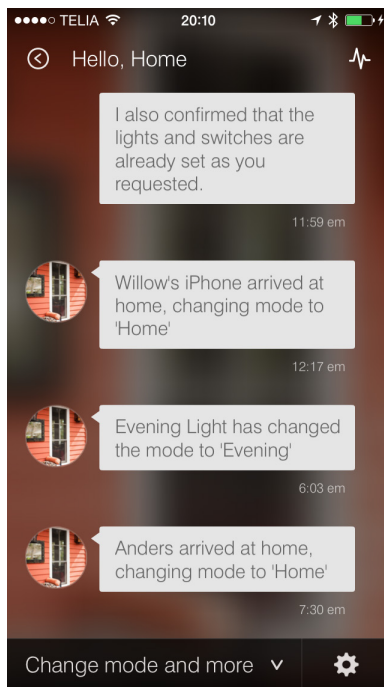
Where you can, link alerts to the means to act on them. For example, “you left home but the heating is on, do you want to turn it off?”

## **USER MANAGEMENT**

If your system may potentially be used by more than one person, you may need to provide ways to identify them or allocate permissions to access certain functionality. Or the system may simply capture data about different individuals and what they were doing. This could be anything from identifying a particular user on the bathroom scales, who was driving or who turned the heating up, to giving someone the means to unlock the front door and disable the security alarm or

assigning a neighbor as an emergency contact if the flood sensor is triggered. Connected home systems may also track presence—who is in and who is out of the house—using keyfobs, personalized alarm panel codes, or mobile location (see Figure 12-10).

Designers also have to think carefully about how different users are identified. A physical device, such as a keyfob, wearable, or smartphone, may be used as a proxy for the user: if Jim's keyfob just came home, that means Jim just came home. That might work most of the time, but it isn't necessarily accurate. Jim might have lent the fob (and his keys) to someone else. Jim might sometimes let his kids borrow his phone to watch videos. Gaining access to the phone means they may have access to other apps on there, too. Or the phone may be stolen. Can you assume that it's OK for anyone in possession of the user's device to be able to act as the user? If not, how are you going to authenticate them? You could require a PIN code to access certain app functions, but will that make day-to-day interactions feel slow and inefficient?



**FIGURE 12-10**

The SmartThings app shows who has arrived home and when (based on the presence of their smartphone)

Guests and visitors pose additional challenges: they may need limited access to parts of the system, or access for a limited time. For example, a houseguest might need to use the door lock, lighting, heating, and security system for a week. But you might want them to have only limited rights. It might seem easier to grant basic system access to anyone who can get onto the WiFi network, but that might cause security risks (see Chapter 15 for a discussion of security).

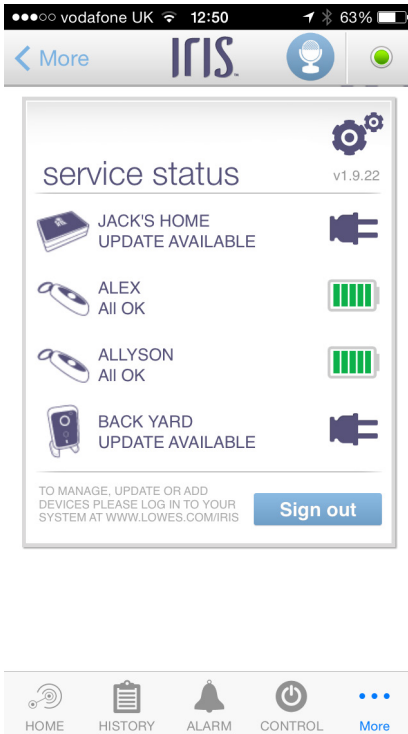
Some users might need to be treated differently—for example, non-residents in a home system. A heating system might track the location of every mobile device on which an app is installed to figure out when the house is likely to need heating. A part-time nanny or housekeeper might need access to the heating app while at work, but his or her location should not be tracked when they are not working.

This can be a sensitive area in which to design. Exposing data about who has been doing what, and allocating (or withholding) permissions to do or access particular things can be a source of friction (see Chapter 11 for a discussion of privacy). In a complex system, it can also cause admin headaches: few of us wish to spend our lives being sysadmins and gatekeepers for our friends and family. The right solution depends enormously on the type of system and context of use. A light touch, balanced with the need for security, is often best.

## **SYSTEM MAINTENANCE**

You will also have to consider how to handle system updates. You may need to deliver firmware or software updates to your devices. Will these be pushed automatically to devices (with the option to roll back if the update causes problems)? Or will users need to agree to them explicitly (see Figure 12-11)? There's potential here for user irritation and confusion as a whole new range of consumer devices suddenly require more active management. The set-top box that requires a 10-minute software update just as you sit down to relax in front of the TV is a new frustration of modern life.

Hardware may be an issue, too. How easily can a device be replaced? Does replacing a spent light bulb require the user to reconfigure all the lighting rules that applied to the old bulb, or can it simply be replaced?



**FIGURE 12-11**

The Lowes Iris Service Status screen shows that firmware updates are available for the gateway and security camera (image: Lowes)

## Discovery

So far in this chapter, we have focused on the experience of setting up and configuring a system that the user owns, perhaps in the home. But in other cases, consumers will be accessing systems in public or other people's spaces. How do they discover the system's presence, what it can do, and understand how to interact?

In a public space, users will need to be notified of the presence of the system in the first place. For example, a sign might inform them that beacons are available to help guide them around the airport (for which they may need to download an app). Or a sign might indicate that an NFC tag is available to provide information linked to a bus stop or point of historical interest (see Figure 12-12).



**FIGURE 12-12**

Sign on a London statue of Rowland Hill, postal reformer, indicating that content can be accessed via NFC tag

Perhaps users are aware that the system is present, but need guidance on how to interact. This will become more and more common the more devices in the environment are connected. The bus stop that broadcasts a web URL for bus times (e.g., the Physical Web project outlined in Chapter 2) needs some visible way of indicating that it can be interacted with.

All of these examples require clear signposting. Until this technology becomes familiar and everyday, service providers will need to indicate not just that a system is available, but also how to interact with it. This will need clear affordances, and establishing a shared vocabulary for explaining interactions to users (see Figure 12-13).

## Control Experiences

In this section, we'll look at some common approaches to designing controls: the actions that users can take with the system. How can you organize controls so that the actions users most need are quick to access, how do you design the experience of sending commands to other devices, and how can you ensure users always know what's been happening on the system?



**FIGURE 12-13**

Oyster travelcard readers are clearly signposted in London rail and tube stations; Oyster created the vocabulary of “touch in/touch out,” which has become standard London vernacular

## **PRIORITIZING KEY CONTROLS**

If you’re offering controls to users, or any interactions in which they will expect a response from the immediate environment, speed is crucial. The speed with which the system responds is part of this, but the inherent challenges of latency and synchronization (discussed in Chapters 3 and 9) can make this tricky. But it’s also the speed of getting to the controls in the first place, such as locating and opening an app and tapping several times to find the right command. As Gawain Edwards puts it, “The best way to get a user to interact is to make it faster to use a smartphone than it would be to press controls on a device directly.” In a system they use frequently, users need very rapid access to the most commonly used controls, even if that makes it slower to access less common controls. You can automate this by guessing at the most used functions (see Figure 12-14), but in practice this tends to make UI elements move around.

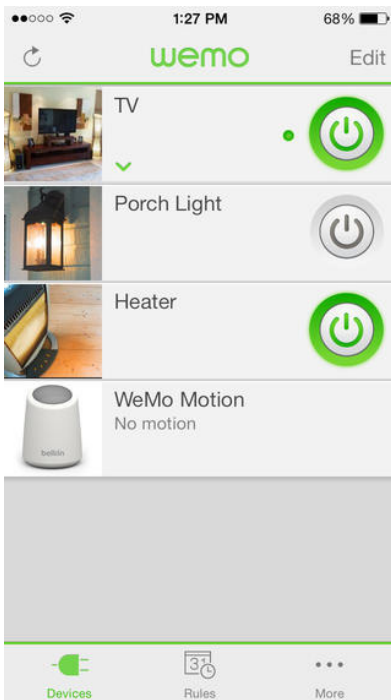


**FIGURE 12-14**

Revolv (now acquired by Nest) tried to make guesses at most used functions on the dashboard (image: Revolv)

This can paradoxically make it harder to find the right controls as users can't rely on motor memory to find them. Or you can allow users to customize their own top controls, but this creates additional admin overhead and may not be used.

Dashboards, composed of key devices and some basic controls, have become a common design solution, particularly in connected home systems. There is currently no consistent dashboard pattern: some simply display all devices and users have to navigate into each one to interact, while others surface basic interaction options right on the dashboard. The WeMo example in Figure 12-15 includes on/off switches for some of the devices listed.

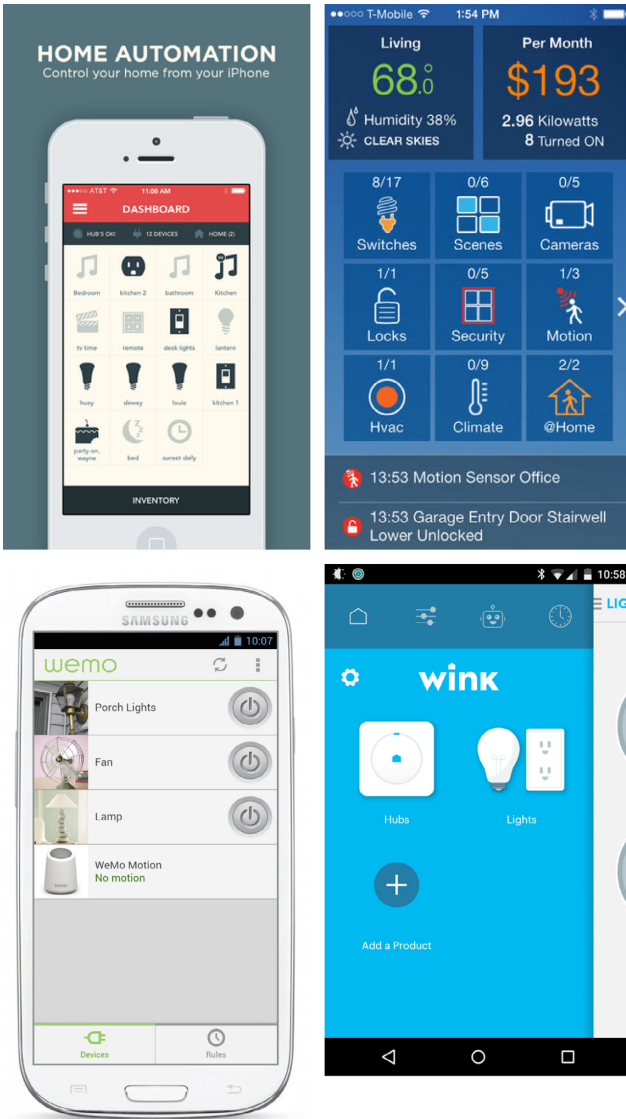


**FIGURE 12-15**

The Belkin WeMo app lets users add a name and a photo of the respective device to include in the overview (image: Belkin)

In a system of relatively few devices and controls, dashboards can be a useful way of providing access to key functionality (see Figure 12-16). But they don't always scale up very well. Imagine a dashboard for tens or even hundreds of devices in a house: it would be more like a flight deck. Some people might like the idea of a flight deck for the home, but they are not mainstream consumers. Dashboards are a general-purpose design solution, and can lack the clear focus that helps explain complexity to users. For mobile UX, in particular, a more focused approach tends to work better.



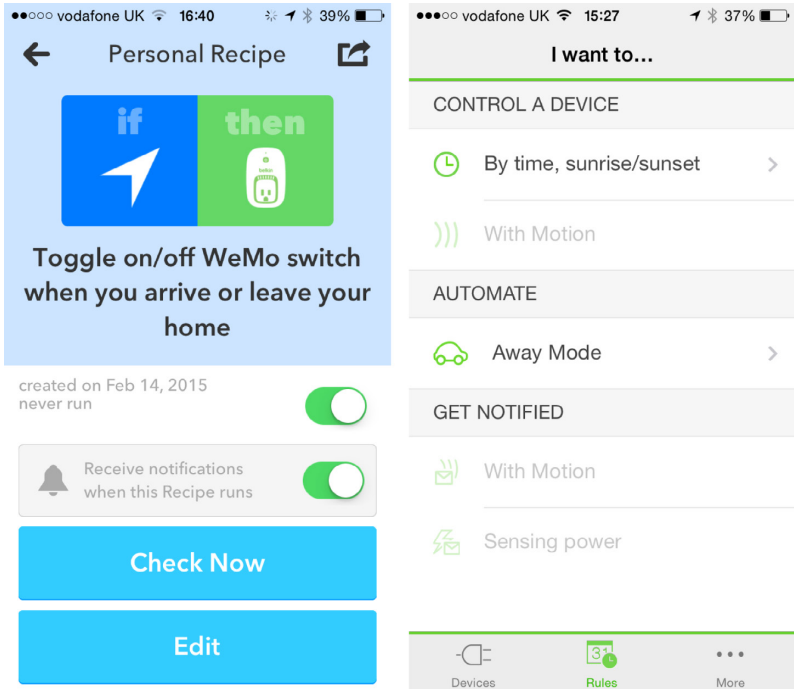


**FIGURE 12-16**

Different smartphone UIs showing dashboards of multiple connected devices (clockwise from top left: Revolv, Simplewave for Vera, Wink, WeMo)

## GROUPING CONTROLS

Connected product systems often provide several ways for users to aggregate and automate system functionality. For example, “rules” can be used to create automated system behaviors, often using “if... then...” conditions, such as: “if motion is detected turn on the light” or “if no one is at home turn down the heating” (see Figure 12-17).



**FIGURE 12-17**

Smartphone UIs that allows users to automate behaviors (left: IFTTT, right: Belkin WeMo Switch)

These programming-like experiences are appealing for early adopters but can be very challenging for consumers. We discuss their pros and cons in more depth in Chapter 15.

Connected home systems, perhaps the most complex of the current consumer products, have several conventions for grouping devices and functionality into commonly used patterns, such as modes (e.g., vacation, night) and rooms or scenes (such as lighting settings for dinner parties or movie watching). These are discussed in more detail in Chapter 15.

## SENDING COMMANDS TO OTHER DEVICES

Controlling one device from another is not quite as straightforward as controlling actions on a single device. In addition to the interusability considerations discussed in Chapter 9, there are a few additional quirks of which you may need to be aware.

First, you may want to make sure that the user has finished interacting before you send the command across the system. This is particularly the case if the user is adjusting the value of a setting (like a heating temperature or lighting level) rather than just turning something on and off, and if the interaction is going via your Internet service. Actions like this can take multiple taps or key presses. Passing updated instructions after every tap will create a heavy load on the system (especially if hundreds or thousands of users are doing the same thing). It may be better to buffer instructions and wait until you're certain the user has finished interacting before passing the command to avoid overloading the network/system. For example, adjusting the temperature via the British Gas Hive app involves multiple taps and drags on the target temperature circle (see Figure 12-18). As the user changes the temperature, the circle flashes between the color of the original target and the new setting. Once the user has finished interacting, the command is sent to the system via the Internet and the phone vibrates to confirm this.

Second, consider how confident you can be that the action on the control device has produced the desired effect, and whether it's necessary to give the user additional feedback. For example, you might turn on a smartplug attached to a lamp, intending to turn on a light. But if the lamp's own switch has been turned off, or the lamp has been disconnected, the smartplug may be powered but the lamp won't be on. In this case, it might be useful to tell the user that despite the plug being turned on, no current is being drawn.

## ACTIVITY FEEDS

When a system is partly controlled by autonomous or user-automated behaviors, it's important that there's a way of viewing what the product did when, and why it did it. If it behaved in an unexpected way, users need to know why, and what to change if needed. Activity feeds, or event logs, list system events and explain why they happened.



**FIGURE 12-18**

The British Gas Hive app homescreen (image: British Gas)

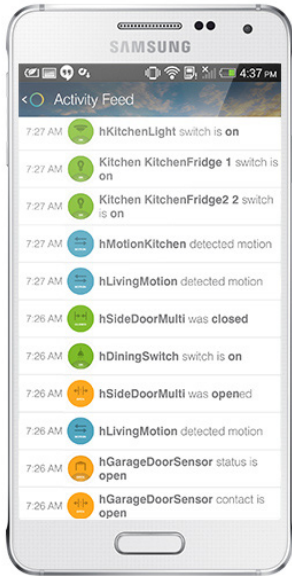
The events shown might have been initiated by:

- A user (e.g., Kelly’s keyfob came home, James turned the heating up to 72°F)
- An automated rule (e.g., the hall light was turned on by the evening security rule)
- An autonomous decision (e.g., the dishwasher was paused to balance demand on the electricity grid)

In Chapter 15, we discuss activity feeds in more depth in the context of automation and autonomous systems. Figures 12-19 and 12-20 show some examples of activity feeds from current live systems.

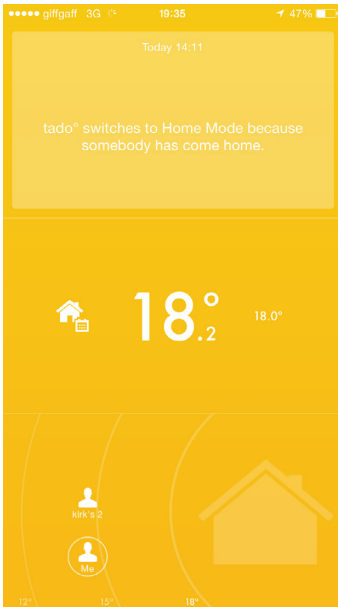
## Platforms

A software platform, as we saw in Chapter 3, is a type of software framework that can be used to build multiple remote service applications. Platform software usually runs in the Internet service (and perhaps gateway, if you have one) and controls how devices and software services work together.



**FIGURE 12-19**

The SmartThings activity feed (image: SmartThings)



**FIGURE 12-20**

Tado displays recent activity on the homescreen

Platforms provide the common functionality that all the applications or services running on top will need: such as getting devices onto the system, configuring them to work together, handling user accounts. This can help to ensure a consistent experience.

In a simpler system, the platform may provide direct access to device data or controls through APIs, so the frontend UI is talking more or less directly to the devices themselves. In a more complex system, there may be *middleware* that provides an abstracted view of system components and specifies some logic about how they should work together. This gives you more flexible and efficient ways of organizing and coordinating your applications and devices as your system grows.

If you're making a very simple system, you may not yet need to think about platform design issues. However, as you add more devices and functions, you may very soon feel the need to think about creating a coherent experience across multiple devices and end user applications.

The UX design work that goes into a platform covers two key areas:

- Provide common design components that help create consistency and coherence in the UX across different functions and applications
- Data models and logic that describe how all the components in the system relate. These can be used to organize controls, automate system functions, and interpret data in more meaningful ways.

The first need you'll run into is for common design components. We'll cover those in this chapter. You'll need to think about data models and logic as your system scales and becomes more complex. We'll cover those in Chapter 15.

## **COMMON COMPONENTS**

Your platform will need to provide a set of common functional components that support all your devices and services. In UX terms, these will help ensure a consistent experience and make it easier and quicker for you, or other designers, to create applications on your platform. These components will probably include basic admin functionality, such as adding and removing devices and perhaps managing users (covered in “In-Life Housekeeping”). Next, we'll consider some additional components for more complex systems, as well as ways of making the overall UX design feel coherent.

## **A common design language for UI elements**

The more complex the system, and the more controls there are, the more important it is to establish a common design language. This will help ensure an appropriate level of consistency throughout the design. For example, you may need common ways to represent controls or states such as on/off, up/down, less/more, timer or schedule/manual, hot/cold, set/unset, locked/unlocked. You may also need common ways to handle notifications: like emergency alarms, low-battery notifications, text messages, app notifications (of various types, depending on mobile platform), emails, outbound calls, and status information.

## **Design templates and patterns**

You may need to create a variety of different UX templates or patterns (structures that repeat) in the design. They might enable you to support different types of devices, organize functionality in different ways, or provide common ways of handling behaviors that are similar. For example, that might include devices (e.g., a light, a radiator control), services (e.g., the security alarm), activities (e.g., watching a film), insights (e.g., you're wasting energy running the heating when no one's home), people (e.g., Jimmy is home from school), or location (e.g., the living room is 72 degrees with four lights on, and Jimmy is there playing on the Xbox).

You probably won't need all of these (if you do, you're dealing with a very complex system that would benefit from the kinds of data models described in Chapter 15). But you might need a mixture of approaches. If your system supports multiple applications, templates and patterns are also an easy way to ensure they are presented in consistent ways. The more complex your system, the more important templates become. Your system might support several different types of lighting, all with slightly different features—for example, dimming, color changing, or the ability to be controlled alongside other lights as part of “scenes.” But the screens and user interactions for controlling all those lights need to feel like a family. The aesthetic styling, layout, and behaviors need to be appropriately consistent: things that work the same should look and behave the same, any differences should be due to the unique capabilities of a particular light. (You might even need to have a screen to handle a generic device: a Z-Wave gateway will control any Z-Wave device to a basic level, even if it's just to provide an on/off switch. Users might use your system to control devices you haven't even thought of.)

Another challenge to consider for designing templates is scale. If users may have anything between one or two devices up to 100, does the UX design need to adapt, and if so, how?

### **Handling instructions from multiple users/devices**

If a system has multiple users, you'll need to consider what happens if they are trying to send contradictory instructions at the same time. One person may try to turn the heating up as someone else turns it down. One person may try to turn a light on from a smartphone app as the other turns it off from a physical switch, making it flash on briefly then off again immediately. Deciding which takes priority can be tricky or even impossible. This makes it important to design things that go to set states (i.e., on, off, etc.) rather than just a toggle switch. If you are trying to turn a light off with a toggle switch, but someone else has already turned it off (without your knowledge), the toggle switch will just turn it on again.

Conflicting controls can also come from automated rules, or autonomous systems. You may be turning the TV or a light on, but a rule says it ought to be turned off after 11 p.m. What happens then? Does the system defer to the user, or insist on turning it off? Would you want to enforce the rule for a TV in a child's bedroom, but allow overrides for the landing light? What if the child is ill, can't sleep, and the parent wants to turn it on? (In Chapter 15, we discuss the challenges of designing automation and "programming-like" experiences in more depth.)

### **Viewing interrelationships between devices, services, and rules**

If you have many devices, controlled by different services or applications, users will need a way of viewing these interrelationships. That might mean, for example, a way of viewing information about a device, showing which services it is used with, which rules it's being controlled by, which other devices it may be controlling or controlled by, what they do, and any data it reports (and to where). If you have multiple users on the system, you might even want to see and control who has permission to use it, or to see its data.



## Automation

Users need a central way to configure, view, and manage automation and the platform can deliver this. In UX terms, this will involve an interface for configuring them, clear visibility of any rules that are running and next scheduled actions and an event log that explains what has recently happened and why. You might even want proactively to suggest ways that devices can work together.

Designing for automation means asking users to engage in program-like activities. This can be extremely challenging to get right. We discuss this in more detail in Chapter 15.

## The Technology of Getting Things Connected

In this section, we'll dive into some technical requirements for connecting and pairing devices. If you're designing a setup process, these will determine chunks of the interaction flow. Bluetooth in particular supports a number of alternative pairing patterns.

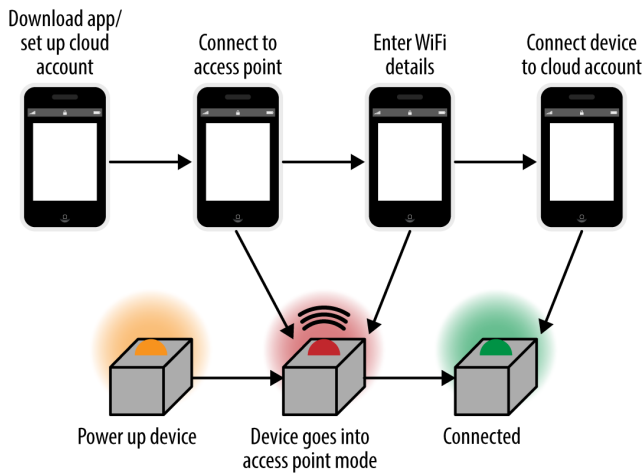
### SET UP GATEWAY OR DIRECTLY INTERNET-CONNECTED DEVICE

This section deals with connecting up devices that connect directly to the Internet over Ethernet or WiFi. This includes dedicated gateways (which provide connectivity to low-powered edge devices), and edge devices that connect directly. We describe the process of setting up gateways later, but the process is much the same.

To set up a gateway, the user must power up the gateway, connect it to the Internet, and associate it with a user account on the provider's Internet service. The gateway may or may not work without the Internet service account.

Gateways are often connected via an Ethernet cable to a broadband router, or WiFi. It's easy to connect a gateway to a broadband router using an Ethernet cable. Many connected home gateways are "headless." That might conjure up images of ghostly horsemen, but it just means they don't have an integral UI. Where the gateway needs to connect to a WiFi network and does not have Ethernet, the process is more complex, as there's no "onboard" way to enter the desired WiFi network's SSID and password. These devices often have a WiFi radio that can function in two modes. On first use (or reset), when it doesn't have

any network details stored, it functions as an access point. The user connects to this local network to configure the WiFi settings on the device (see Figures 12-21 and 12-22).



**FIGURE 12-21**

Configuring a device that acts as an access point



**FIGURE 12-22**

The Ninja Sphere gateway in access point mode; the LED image shown on top indicates that the gateway is waiting for the user to connect their smartphone to the Sphere's temporary network to configure the WiFi (image: Ninja Blocks)

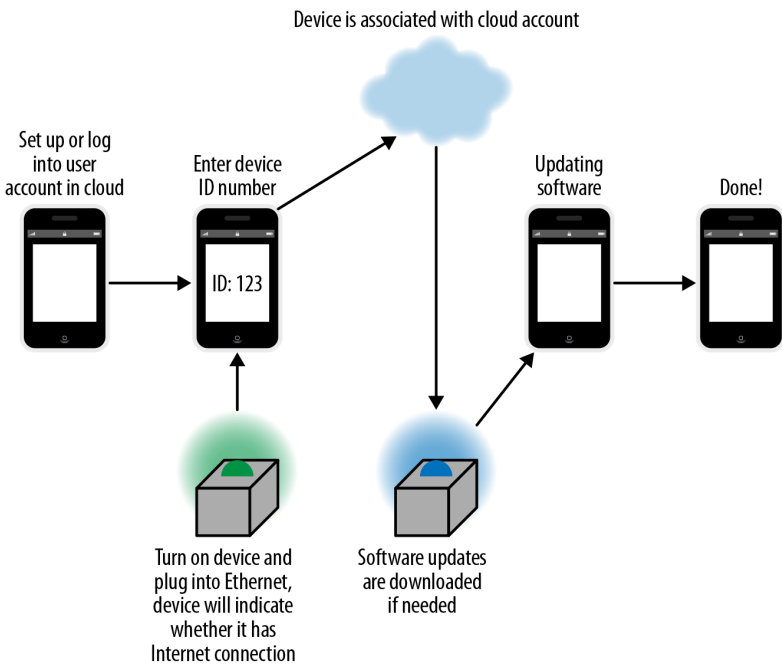
The setup process can be device-led: the user powers up and connects the gateway or device then has the option of associating it with an account on an Internet service. Or it can be service-led: the user logs into an Internet service account first, then associates the gateway or device with the account.

Device-led setup can be useful when the system could be used offline (i.e., the gateway is connecting local devices to each other and can function without the Internet). If the user has bought your device “off the shelf,” it may seem more logical to get the hardware up and running first rather than start by setting up an account.

Service-led setup may be simpler for end users, especially for a headless gateway and/or if an operator wishes to provision apps and update software/firmware on setup. Service-led setup allows accounts to be set up in advance by service provider, useful for complex services or subscription models.

Even if the user doesn’t need an account at first, you might still want them to use a mobile or web app as part of setup. This might be the only way to give the correct WiFi details to a headless gateway (as we’ll explain later).

Figure 12-23 shows an example of service-led setup for a headless Ethernet device.



**FIGURE 12-23**

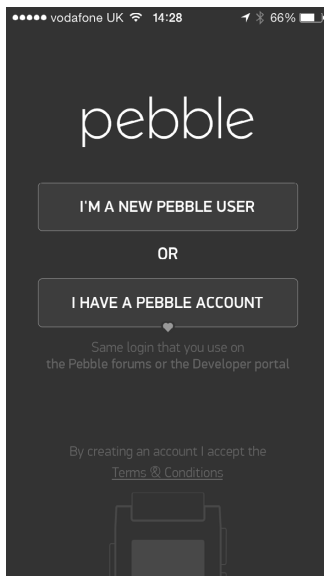
A service-led (cloud-based) flow for a headless Ethernet device

Using a web or mobile app UI, the user sets up and/or logs into a preexisting user account on the Internet service. They plug the gateway into the router and turn on the power. The hub will have some way of indicating whether it has an Internet connection, such as an LED.

In the web/mobile app, they need to enter an ID number, which is printed on the hub or supplied in the box (as an alternative, they could scan a QR code or NFC tag on the device).

The hub is associated with their account on the Internet service. The service will then check the gateway's firmware/software version and perform any updates needed. It may also download application software to the gateway to enable particular functionality according to the user's profile (for example, they may have subscribed to a home security package).

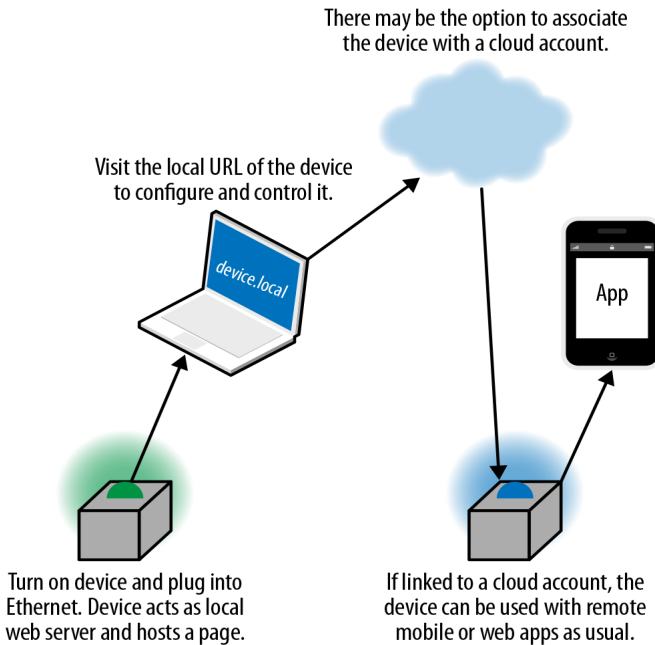
Pebble (see Figure 12-24) is an example of a service-led setup. It requires the user to log into or create a cloud account before connecting up devices.



**FIGURE 12-24**

Pebble setup is service led: the user sets up an account before configuring the device

It's much less likely that a product would require a consumer user to set up a headless Ethernet device without a cloud account first, as it's considerably more complicated. But this is the model by which broadband routers are generally configured, and it's possible you might encounter it. Figure 12-25 shows how this works.



**FIGURE 12-25**

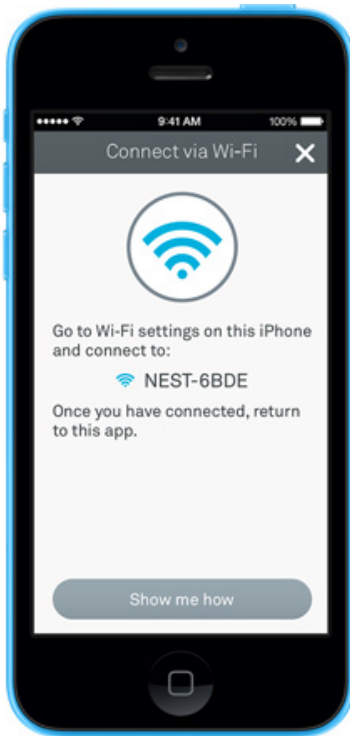
Setting up a headless Ethernet device that does not require a cloud account

The user needs to plug the gateway into the router and turn on the power:

- Using a web browser on a mobile device or computer, they visit the local URL of the gateway, which should show a login screen.
- From this local web interface, the user can configure and perhaps also control the device.
- If the device needs to be accessed on the Internet, the user can then set up and/or log into an Internet service account. This associates the gateway with their account.
- From then on, they can use the “normal” mobile or web app to control the device.

Setting up a headless WiFi device is more complicated. The user needs to provide the device with the WiFi network name (SSID) and password, but the device doesn't have a screen or keyboard by which to enter these. A common approach, as described previously, is for a device that has no WiFi network configured to act as an ad hoc WiFi hotspot (refer back to Figure 12-21).

This is how products such as the Nest Protect and Belkin WeMo Switch work (see Figure 12-26).



**FIGURE 12-26** Getting the Nest Protect onto WiFi (image: Nest)

Switching WiFi networks on your smartphone or PC is fiddly, and takes the user outside the areas of UX design that you control. It may be a good idea to supply printed instructions for this phase, as the user will be leaving your app and may forget what to do next or get lost. This is especially risky on a mobile device where they will have to switch between your app and their phone's settings to read and execute instructions.

You could potentially put the WiFi details in a named file on a USB stick and stick this in the gateway, but the gateway firmware would be configured to expect a file with exactly the right name and the contents would have to be formatted exactly right (see Figure 12-27). This is fine for expert users but isn't a very friendly option for mass-market consumers.



**FIGURE 12-27**

Adrian McEwen and Russell Davies's Soundbox project (which experiments with the use of sound as an ambient notification channel) allows the user to add WiFi network information as well as software upgrades via a USB stick. The system was based around a Chumby embedded computer (Chumby Ltd is now defunct). The Chumby would find the right files for network configuration automatically from the USB stick (image: Russell Davies).

Push-button configuration is a standard supported by WPS (WiFi Protected Setup) that theoretically makes it easier to add devices to a network. Pressing a button on the WiFi router and on a compatible device should get them connected. For example, the Samsung SCX-3405W printer can be connected to a network by first pressing the button on the access point, then pressing a series of buttons on the printer itself (see Figure 12-28).

Some systems attempt to get round the challenge of getting WiFi details into a headless device by using “out of band” methods to transmit the details. For example, the Electric Imp uses a method called

BlinkUp (described in Chapter 8) to convert WiFi details stored on a smartphone to a series of flashes of the phone screen. A light sensor on the imp “reads” the flashes.



**FIGURE 12-28**

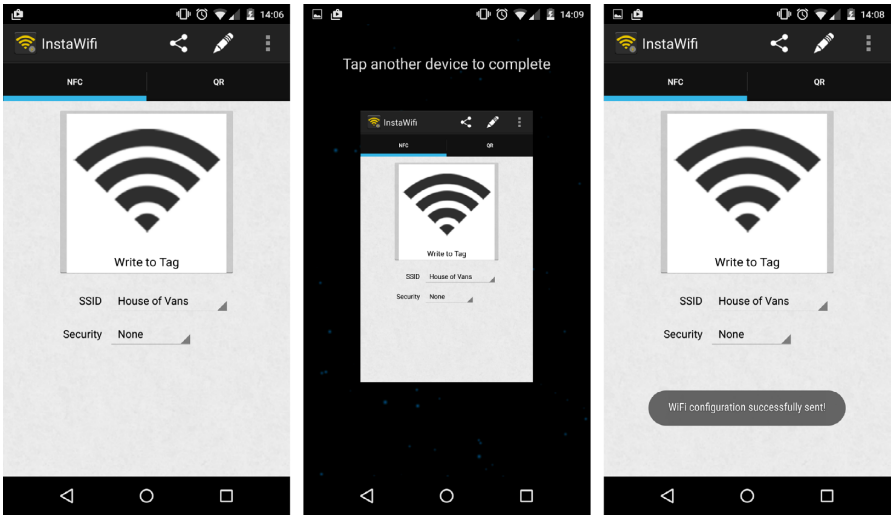
The Samsung SCX-3405W laser printer has a WPS button, circled here (image via Samsung.com)

Standards are in development for out of band (OOB) pairing for WiFi over NFC, but at the time of writing are not widely available in consumer products. The WiFi pairing through the NFC Android app is an early instance (see Figure 12-29). This allows users to send WiFi network details over NFC to another device with one tap, and could be of particular use in setting up devices with limited input mechanisms, such as printers or other embedded devices. OOB pairing using NFC is discussed in more detail in “Bluetooth pairing options.”

Even if your device lacks a screen, it is still useful to use LEDs or other means to communicate status information about connectivity, during setup and afterwards. States you might need to convey include:

- Device has no network access/credentials that need setting up (configuration mode)
- Attempting to connect to network/Internet service
- Successfully connected to network/Internet service





**FIGURE 12-29**  
Android devices sharing WiFi details over NFC

- Error connecting to network/Internet service (e.g., the service might be unavailable or the network might be down)
- Gateway/hub devices may also need to indicate whether local connectivity to edge devices (e.g., Bluetooth, ZigBee) is working OK or not

The Wink HUB uses different color and blink patterns to show the current status of the hub (see Figure 12-30). The BRCK case study in between Chapters 12 and 13 also describes the use of different LED color and blink patterns to show connectivity and battery status.

Once the gateway is set up, you can move on to pairing edge devices with the gateway.

## MOBILE DEVICES

Setting up a device or gateway to use mobile data is much easier. If the device ships with a SIM card, the user may not need to do anything at all to get it connected other than turn it on, and it may already be connected to their account.



**FIGURE 12-30**

The light on the front of the Wink HUB is used to show the current status of the HUB. Flashing pink means it is disconnected from the WiFi network and waiting for credentials (configuration mode). Flashing yellow means it is attempting to connect to your WiFi network and the Wink Internet service. Solid blue means it is online and connected to Wink. Other color/flash patterns are used to indicate when a device is being paired or removed, successful pairing or problems with pairing, and changes in state of connected devices (e.g., a light has been turned on).

Using mobile data can be an easy option for setup: at most, the user has to insert a SIM card, and many devices may ship with the SIM pre-installed. The network operator handles provisioning. But it is an expensive option: either the service provider or the user will have to pay for mobile data. Good Night Lamp chose to supply their initial 100 run of device pairs with a five-year mobile data subscription as customers wanted to give lamps to elderly relatives who might not have WiFi or be confident setting up devices (see Figure 12-31).



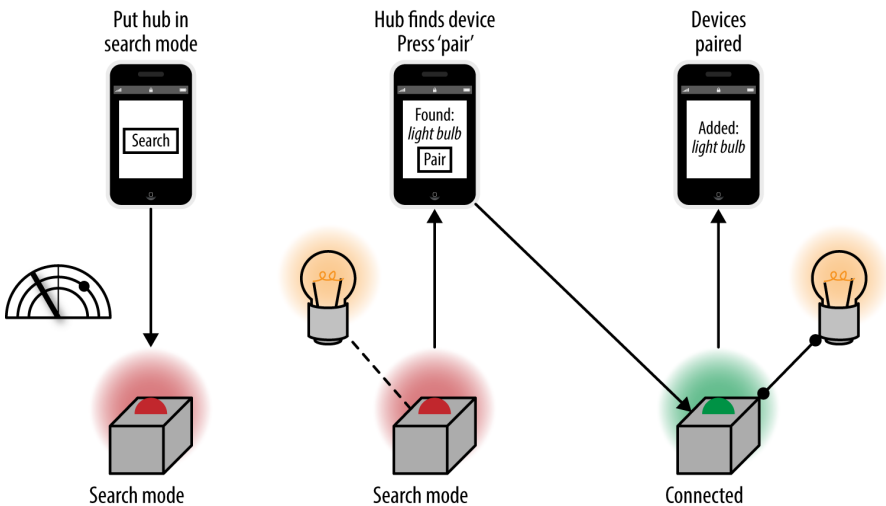
**FIGURE 12-31**

The Good Night Lamp ships with an integral SIM card; it requires no setup to get online and works anywhere in the world (image: [goodnightlamp.com](http://goodnightlamp.com))

## PAIRING DEVICES

“Pairing” is the action of creating a shared security association between two devices over a local network, such as ZigBee, ZWave, or Bluetooth. In other words, you are telling them to trust each other (for certain, possibly limited, purposes). This allows them to share data with each other, or trigger functionality. This may mean pairing an edge device with a gateway, or with another edge device.

In general, pairing for the first time requires explicit user interaction. One device is instructed to search for other nearby devices (aka it is in “search,” “discovery,” or “acquire” mode). The other is set to a discoverable mode in which it broadcasts its identity and basic capabilities to any searching device that asks. If one device is a gateway, this will generally be the one doing the searching. Some devices (particularly those with no onboard UI) will be discoverable by default, and others (such as a smartphone) may need to be set to be discoverable. See Figure 12-32 for a typical device pairing flow.



**FIGURE 12-32**

A typical device pairing flow for ZigBee/ZWave

Where devices have very limited UIs, a device may pair to any other device that comes within range while it is in search mode. The SureFlap cat flap (see Figure 12-33) can be programmed to open only for “authorized” cats, as identified by the pet ID microchips (passive RFID tags) embedded in the cats’ necks. Authorized microchips are paired to the

flap by putting the flap in search mode and holding the cat near the reader. When all cats are added to the system, the flap is taken out of search mode.



**FIGURE 12-33**

Pairing a cat with the SureFlap cat flap. This process can take some time and effort if the cat has a forceful personality. During this time there is the slight risk that an “unauthorized” cat will attempt to enter the flap while it is in search mode, unwittingly pairing and authorizing themselves while you are chasing your own cat around the house.

If the system has a screen UI, the user may need to choose from a list which devices should pair. Some additional method of authentication, such as entering or comparing PINs (see “Bluetooth pairing options”), may be needed. The user may need to give a device a name or configure it to work with particular system functionality (see Figure 12-34).

Once paired, the devices may be able to connect to each other automatically in the future when in range, or this may require explicit user interaction each time.

## adding your accessories

Please power up your accessories. Put in the batteries, removing any tabs. Plug in any that need external power. (If you are adding a Camera, click [here](#) for instructions)

Each accessory should blink a light to show it is searching for the Hub. When it finds the Hub, the flashing will stop, and a picture will appear below.



When all have stopped flashing and you see them on this page click [here](#) to continue.

**FIGURE 12-34**

Pairing devices with the hub in the AlertMe system (image: AlertMe)

Devices are usually paired one by one, but if there are a lot of devices to be paired (for example, security alarm sensors) they could all be paired in one go. The searching device (in this case, usually a gateway) would be put in search mode while multiple edge devices are powered up, devices would then appear in a list as they are discovered and can be authorized from the list. If there are several devices of the same type, the user will need some way of identifying them to figure out which is which. This might be an “identify” button that flashes an LED, as used on the Lowes Iris smart home system (refer back to Figure 12-9).

Network types such as Bluetooth, ZigBee, and ZWave use device or application profiles to make pairing easier. These are software standards in the application layer of the network (see Chapter 3). They define a set of standard device/application types, and how these should communicate. Devices that support the same profiles will work with one another. For example, in Bluetooth, the Human Interface Device profile is used by mice and keyboards, the Hands Free profile is used for wireless headsets, and the SIM access profile allows devices such as car phones with GSM transceivers to use a mobile phone’s SIM to connect to GSM networks. The technology of profiles is a little different for ZigBee, but in terms of UX the outcome is similar. For example, the Home Automation profile contains a bunch of elements that determine

how controls, lighting, HVAC, sensors, and security should communicate, and descriptions of the types of device that are expected. Devices that fall within the expected types should pair and work easily.

In general, good design for pairing experiences should:

- Allow plenty of time: if discoverability or other device states timeout, ensure that the user has plenty of time to complete their task before the timeout. Bear in mind that they may be slow if they are unfamiliar with the process.
- Minimize the amount of thought required, such as memorizing information between screens or devices, remembering long number or character strings, and inventing keys, codes, or passwords (users tend to use the same ones over and over again so they are unlikely to be unique).
- Minimize the number of key presses or interactions, especially for devices with limited input capabilities.
- Provide good feedback so it's always clear what's happening. For example, the Wink HUB LED flashes blue when pairing or removing a device, then displays a long green or red flash to indicate whether pairing has been successful or not.
- Ensure adequate security: help users ensure they are connecting to the correct devices, and ensure sensitive data exchanges are encrypted.

Pairing devices requires explicit user interactions, because this is currently the best way to ensure that devices trust only the devices they ought to trust. But this may become a burden the more devices the user has. Some researchers are looking at automated methods of detecting and pairing with trustworthy devices. For example, a 2014 study by Markus Miettinen and colleagues investigated ways to detect trusted devices that are kept together by comparing the contextual data measured by their sensors (light and noise). But this is very early stage research.<sup>5</sup>

---

5 M. Miettinen, N. Asokan, T.D. Nguyen, A.-R. Sadeghi, and M. Sobhani, "Context-Based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices," ACM Conference on Computer and Communications Security, 2014: [http://www.icri-sc.org/fileadmin/user\\_upload/Group\\_TRUST/ccsf43s-miettinen.pdf](http://www.icri-sc.org/fileadmin/user_upload/Group_TRUST/ccsf43s-miettinen.pdf).

## Bluetooth pairing options

Bluetooth offers several alternative options for pairing interactions. The choice of which is most appropriate will depend on the input and output capabilities of the devices you are pairing, and the level of security required.<sup>6</sup>

Pairing two devices should be quite straightforward for the user, if both have compatible “profiles” and support the same version of Bluetooth. (Profiles are specifications for different communication patterns between devices, optimized for different purposes. For example, the Human Interface Device profile provides low-latency links for mice, keyboards, and game controllers. The Heart Rate profile is designed to transmit heart rate data.)

The type of security required is determined by the profile. Some profiles, such as Object Push (which is used to send a vCard between mobile devices), don’t require any authentication or encryption as that would make the UX very inefficient and there’s no real security risk to mitigate.

Once paired, devices can reconnect automatically when in range or require explicit user interaction to do so.

Every device has a unique network address, but it is generally identified by a user-friendly Bluetooth name (e.g., “Pebble Watch”) in search results and lists of paired devices. The default name is usually set by the manufacturer but the user can often change this device name (e.g., to “Claire’s Pebble”) if they wish.

Some devices can only be connected to one other device at a time. While connected to one device, they will not appear in searches or be able to connect to other devices until they disconnect from the first device. For example, peripherals such as smart watches may be paired with both a tablet and a smartphone, but only able to connect to one at a time. However, the tablet and smartphone are defined as hubs, and able to connect to multiple devices. Bluetooth 4 (Smart) allows devices

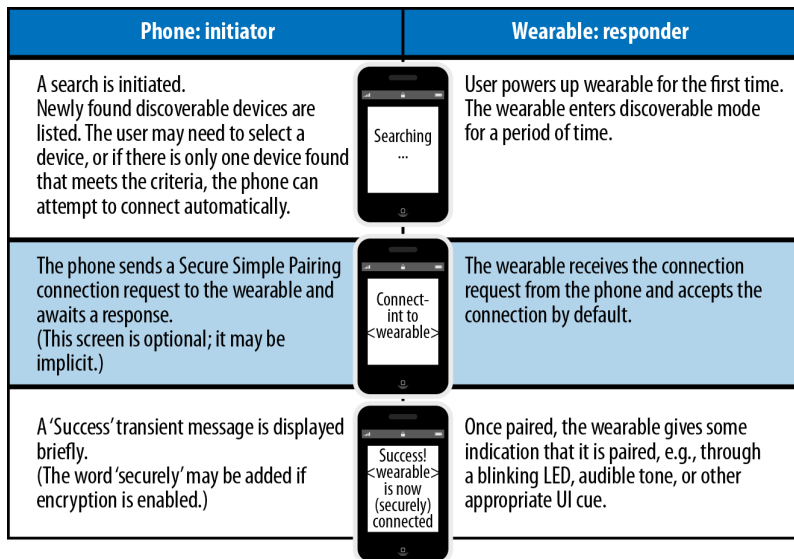
---

<sup>6</sup> The mechanisms described here are those supported by Secure Simple Pairing, available in Bluetooth 2.1 onward, and Bluetooth Smart. For more details, including detailed interaction flows, see “Bluetooth Secure Simple Pairing User Interface Flow Whitepaper,” available from Bluetooth.org ([https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc\\_id=86173](https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=86173)) and <https://developer.bluetooth.org/TechnologyOverview/Pages/LE-Security.aspx> for information on Bluetooth LE pairing.

to act as both peripherals and hubs, so for example a smart watch could act as a hub to gather information from a heart rate monitor, while also connecting as a peripheral to a smartphone, displaying incoming calls and message notifications. (Bluetooth connectivity is discussed in more detail in Chapter 3.)

Beacons (such as iBeacons and Estimote) are Bluetooth devices, but don't need to be paired for normal use. Beacons broadcast small data packets, which the user's smartphone (for example) can use to triangulate their location. The phone does not send any data in return. (See Chapter 2 for more on beacons.)

**“Just works.”** The “just works” approach (see Figure 12-35) is used when one or more of the devices pairing are not capable of displaying (or allowing the user to enter) a six-digit PIN or passing information via another channel (see “Out of band pairing”). This is commonly the case for embedded devices such as activity trackers. The user searches for devices on a screened device (such as a smartphone), selects the desired device from a list of search results, then connects. Pairing is complete.

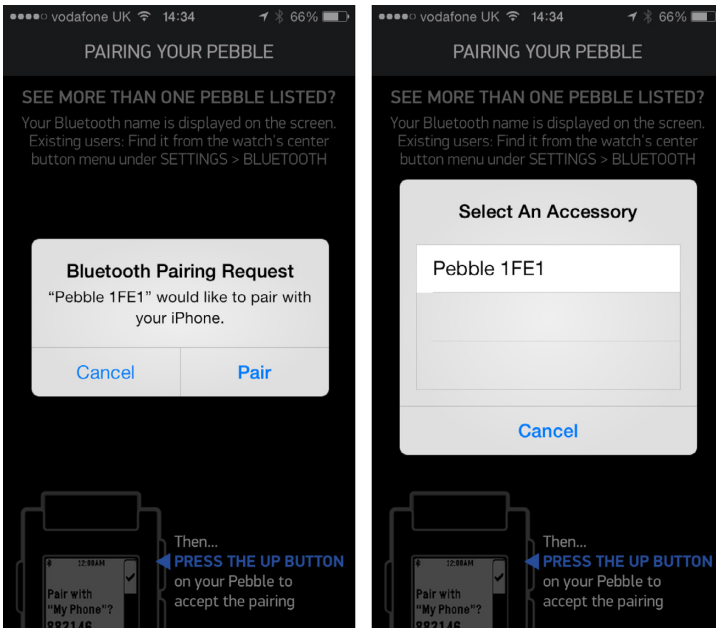


**FIGURE 12-35**  
“Just works” pairing<sup>7</sup>

7 Adapted from “Bluetooth Secure Simple Pairing User Interface Flow Whitepaper,” [https://www.bluetooth.org/docman/handlers/DownloadDoc.aspx?doc\\_id=86173](https://www.bluetooth.org/docman/handlers/DownloadDoc.aspx?doc_id=86173).



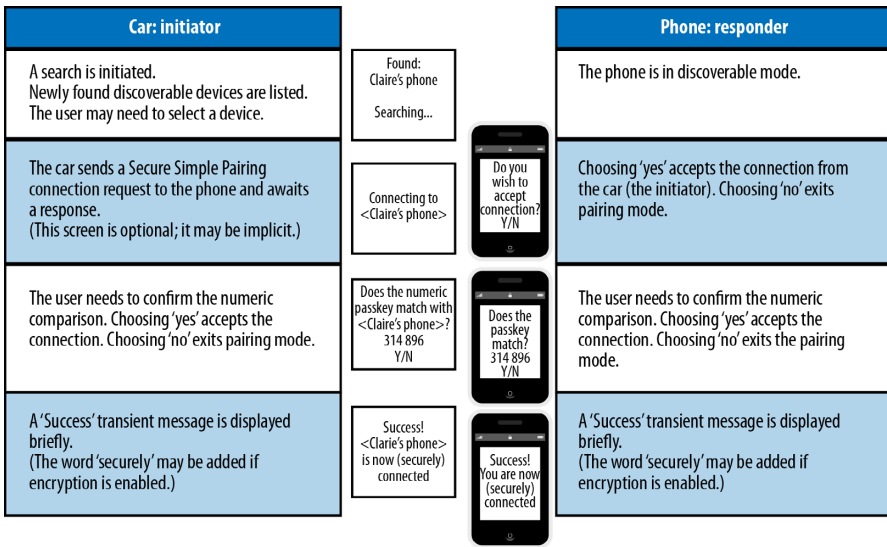
If both devices have displays and some kind of input mechanism, the responding device may ask the user to confirm that they wish to accept the connection from the initiating device. In Figure 12-36, an iPhone has initiated a search for a Pebble watch. To complete the pairing, the user must tap “pair” on the iPhone screen, but also press the up button on the Pebble watch.



**FIGURE 12-36**

The iPhone/Pebble “just works” Bluetooth pairing confirmation

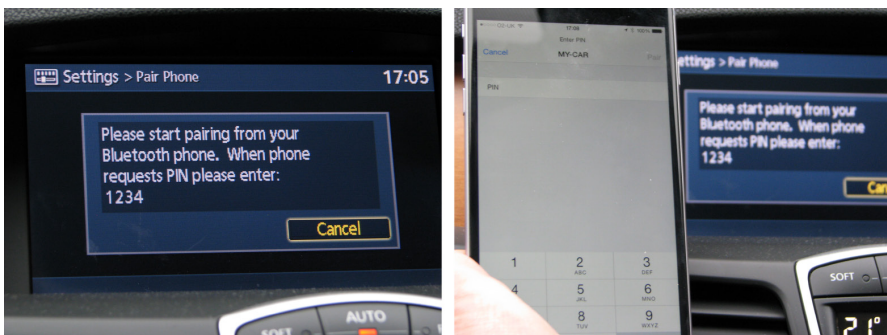
**Numeric comparison.** The numeric comparison approach (see Figure 12-37) can be used when both devices are capable of displaying a numeric passkey. It adds an extra step of security (and where there are multiple devices in the vicinity, helps users ensure that they are pairing the correct ones). As in the “just works” flow, the user searches for discoverable devices from the initiating device (in this example, searching for nearby smartphones from a car), then (if there is more than one) selects the one to which they want to connect. The responding device asks the user to confirm that they wish to accept the connection, then both devices display a six-digit passkey and ask the user to confirm that these are the same.



**FIGURE 12-37**  
Bluetooth numeric comparison pairing

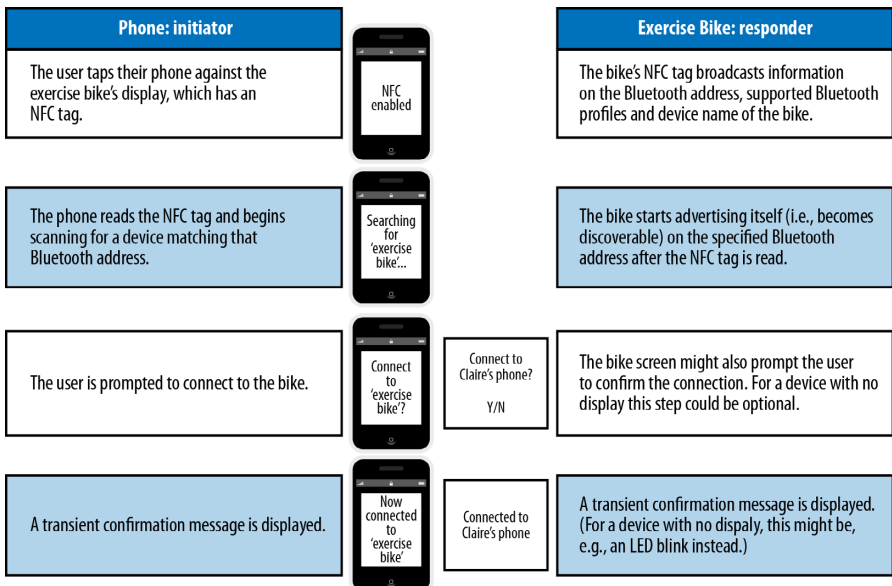
**Passkey entry.** This approach can be used between a device with a display only and one that can accept numeric input, or two devices with numeric input. In the former example, the display shows a six-digit code to the user, which must be entered on the keypad on the other device. In the latter, each user must enter the same six-digit number.

In Figure 12-38, a Renault Laguna car dashboard displays a PIN to be entered on the phone as part of the pairing process. (This is an older standard, using a four-digit PIN. SSP uses a more secure six-digit PIN, but the method is the same.)



**FIGURE 12-38**  
iPhone and Renault passkey entry pairing

**Out of band pairing.** The out of band (OOB) method uses a non-Bluetooth channel (such as NFC) to exchange pairing information (see Figure 12-39). This provides greater security than a “just works” approach, but is simpler for the user than entering a passkey (and suited to devices that may lack keyboards). The user chooses the “add device” option on the initiating device. Both devices must have NFC (or the alternative channel) enabled. Passing the two devices within range causes them to exchange data. They can either automatically be connected, or confirmation may be required from the responding device.



**FIGURE 12-39**

An example of Bluetooth out of band pairing via NFC[1]: a user might pair his phone to an exercise bike in a gym to capture information about his workout<sup>8</sup>

## Summary

IoT products and systems may do all kinds of things. But there is a common set of interactions needed to get and keep them up and running. At least some of these are likely to apply to your system. Some of these,

<sup>8</sup> Adapted from Part 1 of the “Bluetooth Secure Simple Pairing Using NFC” Workshop by Tore Austad of Nordic Semiconductor ASA at NFC World Congress, October 2014: <http://bit.ly/115NsP8>.

like getting devices connected and pairing, may need to be designed around certain technical standards and requirements. For others, there may be more flexibility. There are some common approaches, but the field is not mature enough yet for design patterns. There is definitely scope for innovation!

A good setup experience helps users all the way from getting things connected to getting real value out of the system. There are technical requirements around getting things connected and pairing that will strongly influence this part of the interaction design. You may be able to simplify this, perhaps supply some things preconfigured, and try to provide a flexible process that allows users to complete steps in different order. Ensure the whole experience of setup works coherently, from presales expectation setting to packaging, print instructions, device labeling, and UI guidance. Don't forget to help users configure the system to work the way *they* want, and remember that they may need to come back and reconfigure things later.

All systems will need at least some in-life housekeeping functions, such as device management, notification and message management, user management and system maintenance. You may need to think about ways of organizing controls: what do you prioritize, how do you group them, how do commands flow from one device to another, and how does the user keep on top of everything that's happening? And if your system is public, you'll need to think about how users discover it and understand what to do. There aren't any hard design patterns for these yet, but it's worth being familiar with some common approaches... and then thinking about ways to improve on them.

Platforms are the basis for building systems that support multiple applications. To ensure a consistent and coherent experience across the different parts of your system, you will need to identify and design some common UI components.

## [Case Study 5]

# BRCK: Rugged Portable WiFi Hotspot

BY JEFF MAINA<sup>1</sup>

## Designing the Setup of the BRCK

### WHAT IS BRCK?

“The BRCK is like a backup generator for the Internet.”

The BRCK, shown in Figure CS5-1, is a rugged, self-powered, mobile WiFi device that connects people and things to the Internet in areas of the world with poor infrastructure, all managed via a cloud-based interface. The BRCK was designed and prototyped in Nairobi, Kenya by the team who also created the Ushahidi<sup>2</sup> platform for crowdsourcing crisis information.

Away from the world’s most wired urban areas, connectivity and even power can be unreliable. The Ushahidi team realized that what they really needed was a smart, rugged device that could connect to the Internet any way it could, hop from one network to another, create a hotspot for multiple devices, while plugged in or running on battery power.

The portable BRCK is a small, physically robust device designed to provide Internet connectivity in places where power and reliable data may not always be available (see Figure CS5-1). It can switch seamlessly between Ethernet, WiFi, and 3G/4G mobile phone networks, to provide

---

1 Adapted with permission from Jeff Maina’s blog post “After the User Experience,” <http://www.brck.com/2014/09/after-the-user-experience/>.

2 <http://www.ushahidi.com>

a reliable local WiFi hotspot for up to 20 other devices. The onboard battery can power the device for up to eight hours, or six months in Sleep Mode.

The team also created the BRCK to be an onramp for the Internet of Things. It provides hardware interfaces to connect to external sensors and devices and supports IPV6.



**FIGURE CS5-1**

The BRCK

BRCK is used in a wide variety of settings, to connect both people and sensors to the Internet. Some IoT uses include:

- Enabling remote temperature monitoring of fridges used to store vaccines to ensure they are maintained at safe temperatures even when power is unreliable
- Remote monitoring of the temperature and humidity of warehouses used to store food for distribution by the World Food Programme, to ensure the food stays safe to eat
- Monitoring remote water pumps to ensure they are fixed promptly when they break down

The BRCK's team's motto has always been "if it works in Africa, it will work anywhere."

## **DESIGNING FOR BRCK**

A software UI designer's focus is only on a user's screen. Designing for connected devices is tougher, because designers have to think of the product holistically.

The BRCK is made up of the following:

*Hardware*

The physical BRCK

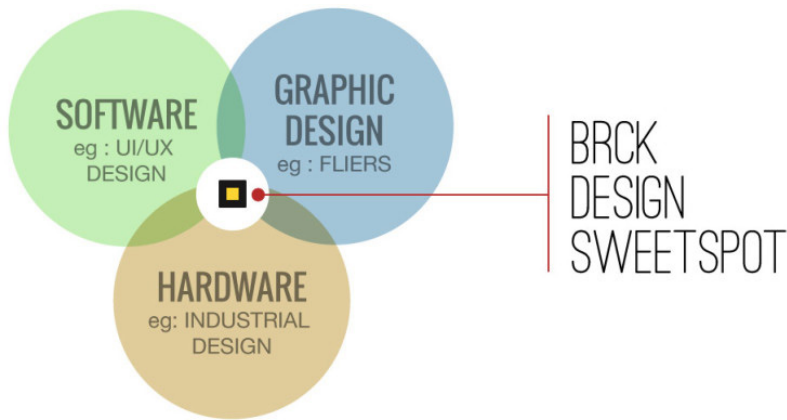
*Software*

Firmware/operating system that sit on the hardware and the cloud used to manage the BRCK remotely via the Internet

*Print*

Packaging box and user guide material

All this has to merge seamlessly so as to ensure the end user gets a product that's easy to use right out of the box (see Figure CS5-2).



**FIGURE CS5-2**

The BRCK UX spans software, print, and hardware design

### USER LEARNING CURVE

All products have a learning curve when it comes to user interaction. Some products may have a steeper learning curve—for example, those that are new in the market or reinventions of what already exists.

It's much simpler to build a new product in a category that already exists. Users are likely to have some expectations about how to interact with it, and designers can build on that mindset.

The BRCK falls into the reinvented bracket. Conventional WiFi routers are designed for markets with reliable mains power and connectivity. The BRCK team is building a product that solves the problem

of Internet connection in emerging markets using a solution that's tailored for those environments. This means unreliable power, power surges, and a failover from "normal" Ethernet Internet to wireless 3G.

We had to look at the typical pain points people face while trying to get online, then design and build a product tailored to those needs. We had to do away with the typical stereotypes of what a router looked like and how they worked, creating a product that would be:

*Mobile*

For when you're mobile

*Versatile*

Made to work where others won't

*Powered (rechargeable battery)*

Thinking through power as much as connectivity

*Extendable*

A platform to build on with software or hardware

*Cloud managed*

Manage your BRCK from anywhere

The BRCK was a product that was designed and functioned totally differently from other WiFi routers. We had to communicate well in order to minimize the learning curve and make it as easy as possible to set one up.

## **THE UNBOXING PROCESS**

A year after we began the process of creating the BRCK, almost all the work was done and the first batch of the BRCKs had been shipped. Our team put a lot of attention in designing and building a product we believe will deliver a great experience. However, as with all other products, there is always something new to be learned once a customer interacts with your product for the first time.

Having interacted with the product for months ourselves, we had clear thoughts about design based on our understanding of how users would interact with what we had built. But we knew we might have to make some minor changes based on the feedback we got after running UX testing sessions at the iHub in Kenya (see Figure CS5-3).





**FIGURE CS5-3**

Testing at the iHub in Kenya

What we learned was that some of the steps during setup that we thought were obvious were not necessarily obvious to end users.

BRCK is a merger of hardware and software, with the setup process explained on a single A4 sheet of paper (the Quick Start Guide—the document in the packaging that explains the steps you take to set up your BRCK for the first time). While the BRCK worked great, the steps for getting the BRCK working, as printed in the Quick Start Guide, had to be thought through again.

This forced us to go back to the drawing board and solve some communication pain points that hindered users in getting their BRCKs up and running in minimal time. We needed to give more detail in the Quick Start Guide, yet not clutter the page with extra information.

### **WHAT WE LEARNED AND CHANGED**

After opening the packaging most people grabbed the BRCK, and only briefly browsed the Quick Start Guide without reading the content. It was only after they couldn't figure out what to do that they picked up the Quick Start Guide again.

Some of the core lessons we learned from the users were:

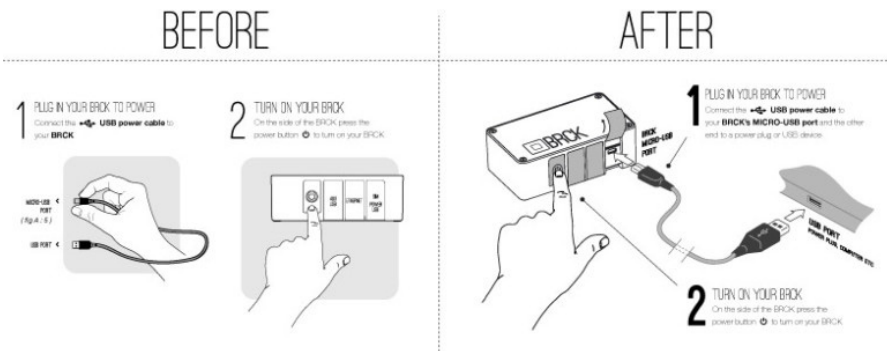
- They weren't sure which side of the micro-USB cable (used to power the BRCK) went where.

- When the BRCK was charging most thought it was ON.
- They weren't sure how long it would take to get the WiFi connection up and running.

After getting this feedback, we set out a goal of minimizing the setup process time. Our goal was for users to be able to unbox their BRCK and fully register it somewhere between 10 to 20 minutes.

To achieve this, we had to reduce the steps and explain each in better detail. This doesn't mean we had to add text: an image can be worth 1,000 words. So we decided to improve the diagrams included in the Quick Start Guide to better illustrate the steps necessary for setting up the BRCK.

We replaced certain graphics and added some in order to achieve this. Figure CS5-4 shows some of the changes we made.



**FIGURE CS5-4**  
Before-and-after visual of the first setup steps

The status light was confusing to users when they first plugged in their BRCK. So we added a graphic to better explain this (Figure CS5-5).

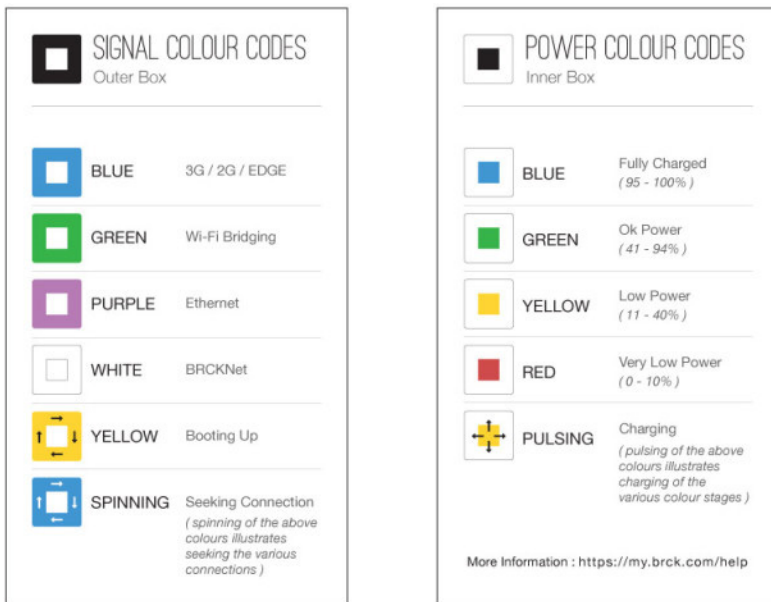


**FIGURE CS5-5**  
Graphic explaining the status light

We then set out to remove the setting of the WiFi from the registration process so as to reduce the steps.

Our greatest challenge was communicating what the status light color codes meant. The top light of the BRCK is the main physical communication point between the hardware and the user. Different states are communicated using different colors spinning and pulsing. The Quick Start Guide was printed in grayscale and we needed a way of conveying this information to our user without using text. We then came up with the color code charts.

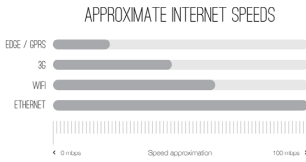
These are business card-sized charts that visually illustrate what each color is and does (Figure CS5-6).



**FIGURE CS5-6**

The color code charts are business card-sized cards which illustrate what each LED state means

Figures CS5-7 and CS5-8 show the full versions of both Quick Start Guides.



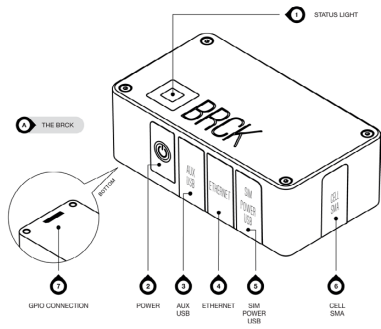
### APPENDIX

- BRCK < NAME >** : The default name of your BRCK, where < NAME > is a randomly generated code for each p.
- Browser** : A software application for presenting resources on the World Wide Web. The major web browsers are Google Chrome, Mozilla Firefox, **Internet Explorer**, Opera, Safari & Lynx.
- Status Light** : The top guada light on your BRCK (Pg A - 7)
- AUX USB** : USB slot to connect external devices eg. phone, tablet etc. (500mA Max (Pg A - 9))
- Ethernet Port** : Ethernet slot used to connect an ethernet cable (RJ45 Cable) to another router (Pg A - 4)
- SIM Port** : SIM card slot used to insert a data enabled SIM card (Pg A - 5)
- Power Micro-USB Port** : Micro-USB slot used to charge the BRCK (use the provided USB cable (Pg A - 5))
- Cell RP-SMA** : antenna expansion port (attach antenna for better coverage) (Pg A - 6)
- GPIO connection** : The bottom port used to connect external devices (Pg A - 7)

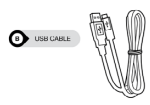
**Warranty** : 6 month warranty against defects in materials and workmanship.  
For more information see <http://www.brck.com/warranty>

## BRCK® QUICK START GUIDE

SETUP YOUR BRCK IN 7 SIMPLE STEPS



### ACCESSORIES



This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:  
 1. This device may not cause harmful interference.  
 2. This device must accept any interference received, including interference that may cause undesired operation.



BRCK © All rights reserved. BRCK is a product of Bahans, Inc.

**FIGURE CS5-7**

The Quick Start Guide before testing and design iteration

## Setting up your first BRCK should take you about 10 minutes to get started, following these 6 steps below

**1 PLUG IN YOUR BRCK TO POWER**  
Connect the **USB power cable** to your **BRCK's MICRO-USB port** and the other end to a power plug or USB device.

**2 TURN ON YOUR BRCK**  
On the side of the BRCK, press the power button **ON** to turn on your BRCK.

**3 CONNECT TO "BRCK(WAVE)" WIFI SIGNAL**  
On your computer, smartphone or tablet search for the BRCK's WIFI signal and connect to it.

**NOTE** - After switching on your BRCK, it will take about **1 minute** for the WIFI signal to be available.

**4 GO TO: <http://my.brck.com> ON A BROWSER** (Chrome, Firefox, Safari, Internet Explorer)

**▲ Not connected?** Do one of the following:

- Plug in an ethernet cable
- Look for another WIFI signal
- Insert a SIM card

Once that is done, ensure you are connected to your BRCK's WIFI, then enter the PIN: <http://192.168.69.1>

**5 SETUP OR LOGIN**  
Setup with a new account or login to an already existing profile.

**▲ NOTE:** Your WIFI Network is not secure. Once you get into your dashboard rename your WIFI Network and Password.

If this is your first BRCK, Setup and now account. If you have an account already, then login to your existing profile.

**6 YAY! YOU ARE CONNECTED**  
Connect your device with your BRCK's WIFI signal (maximum 20)

**▲ Having trouble connecting?**  
Visit: <https://my.brck.com/help>  
You can also visit <http://forums.brck.com/> to meet other BRCK users.

**Note:** The power button is the disconnecting device, and will disconnect all networks from the BRCK.

**FIGURE CS5-8**

The Quick Start Guide after testing and design iteration

# Designing with Data

BY ALFRED LUI AND CLAIRE ROWLAND<sup>1</sup>

## Introduction

The IoT will enable access to vastly more data about the world than we've ever had. By 2020, some analysts estimate that 10% of all digital data will be generated from connected devices.<sup>2</sup> Industrial applications may account for a large chunk of this, but data is also becoming part of the fabric of consumer products. Designers need to understand how to work with information as a “design material.”<sup>3</sup> What can be captured? How should it be interpreted and what insights can be derived? What new products can be developed and what previously unmet needs can now be addressed? And how do we go about designing compelling, valuable, secure, and trustworthy experiences using data? How can we help bring new insights to users through product design, and design products that behave smarter?

This is a rapidly evolving field, covering multiple topics each of which could easily be a book in its own right. In this book, we aim to provide an overview of the key issues.

This chapter introduces:

- The technical context of data in IoT (see page 528)
- Current types of data-driven product (see page 544)
- Considerations for designing with data (see page 550)

---

1 With thanks to Edd Dumbill, Simon Frost, and Christopher Osborne.

2 <http://www.emc.com/leadership/digital-universe/2014iview/internet-of-things.htm>

3 Kuniavsky, *Smart Things*, page 43.

This chapter addresses the following issues:

- How data processing in IoT is often distributed, and what that means for the product (see page 533)
- Typical characteristics of IoT data, and their impact on design (see page 538)
- The different concerns of designing types of data-driven products (see page 546)
- How focusing on users' goals is the key to making data meaningful and actionable (see page 550)
- How users' attention is a scarce resource, and how to be respectful of it (see page 558)
- How products that seek to produce behavioral change need to deliver very specific insights in the right context (see page 565)
- What's different about visualizing data for connected products (see page 569)

## Data in IoT

IoT creates the potential for almost anything that happens in the physical world to be measured (through sensors) or inferred (from the richness of data sources), creating a digital trace. For good or ill, we will know much more about the world around us, and about ourselves (the ethics of what should and should not be captured and what should or should not be done with it are dealt with in Chapter 15).

Data on the level of a single device can be useful, but when those devices are connected, the power of the network comes into play. Not only is device data accessible from anywhere, but information from many devices and many other datasets can be combined. It's interesting to see the energy consumption of a single home. But it's considerably more valuable to understand how it compares with the energy consumption of millions of other homes, correlated with geographical location, demographics, and weather. Data from millions of single devices combines to give us collective intelligence about the world.

Networking allows us to work with data that is distributed across many different sources, but making sense of it can be a challenge. Distributed datasets are often stored in different formats, and are often

unstructured (e.g., free text or multimedia content instead of neatly organized fields in a database). Data scientists working in this field view the growing volume of IoT data as one of the next major challenges. And in practice, poor interoperability (as covered in Chapter 10) may mean that accessing and working with different data sets is a challenge.

In IoT, the bulk of data is likely to come from industrial applications. Examples may include tracking stock (or even livestock), environmental monitoring, and monitoring the health of equipment for predictive maintenance. For example, US rail operator Union Pacific Railroad (Figure 13-1) uses connected sensors to predict and prevent train derailments.



**FIGURE 13-1**

The Union Pacific Railroad uses connected sensors to detect problems with wheel bearings that could cause derailments (image by Emmett Tullos [ravensong75] from Flickr, shared under CC license)

Infrared sensors on the tracks take over 20 million temperature readings of train wheels a day to check for overheating, which indicates impending failure. Trackside microphones listen for noises that indicate damaged bearings in the wheels. On the heaviest freight trains, wheels are even scanned with ultrasound to detect flaws. Fiber-optic cables running alongside the tracks carry these readings back to data centers. There, pattern-matching algorithms identify outlying readings that might spell trouble and notify human experts. Within five minutes of an unusual reading being taken, an expert will decide whether

to take action—for example, notifying the driver to stop the train for inspection or slow it down for safety until it can be repaired at the next station. Using this system, Union Pacific has cut bearing-related derailments by 75%.<sup>4</sup>

The falling cost of electronic sensing and connectivity means that data capture is also becoming a core capability of many consumer products. Electronic components that measure environmental and biometric data such as geolocation, temperature, altitude, skin temperature, and heart rate are readily available, often even in smartphones. Two of the best-known examples of data-driven consumer connected products right now are smart energy meters and health and fitness wearables. But all IoT systems generate and use data to some extent.

Of course, raw data has no value unless you are able to extract meaning from it: in particular, to generate actionable insights. Understanding the data can be interesting. But the real value is derived from what you are able to do differently as a result of the insights. A fitness tracker app that tells you how many more steps you need to walk to offset your calorie intake is providing actionable advice.

Visualization is an important tool for making sense of data. But the output of designing with data need not be a visualization. Data can also be used behind the scenes to design adaptive services that learn about the user, and present the right functionality and content at the right time for that person. IoT enables these services to be used to tailor interactions in the physical environment, as well as through screens. The Disney MagicBand (introduced in Chapter 2; see Figure 13-2) links to an online account as part of the MyMagic+ service. This allows information about a guest's park attraction and restaurant reservations to be combined with their location to personalize the experience. Guests can be greeted by name, pre-ordered food can be prepared as guests approach the restaurant ahead of their reservation, and visitor movements around the park can be used to help Disney suggest itineraries that minimize wait times.

---

4 Chris Murphy, "The Internet of Things," *Information Week*, August 2012.





**FIGURE 13-2**

Guest ID and location information from the MagicBand enables Disney to personalize the guest's experience in the park (image: Ciara Taylor)

Nor is designing with data necessarily about enabling user action. There is a current trend toward services that use data explicitly to promote behavioral change. They may encourage us to exercise more frequently, use less energy, or eat slower (Figure 13-3).



**FIGURE 13-3**

The HAPIfork monitors the user's eating habits and encourages him to eat slowly (image: HAPI.com)

But data can be also be used to enable systems to function more efficiently and even act autonomously. A “smart” thermostat adapts to the home’s occupancy patterns and the occupants’ comfort level. To do this, it gathers and analyzes data, and decides for itself how to act. The product’s developers had to determine:

- What data the thermostat could collect
- How to analyze it
- How best to act on it
- How to keep the user informed of all this and ensure they maintain a degree of control

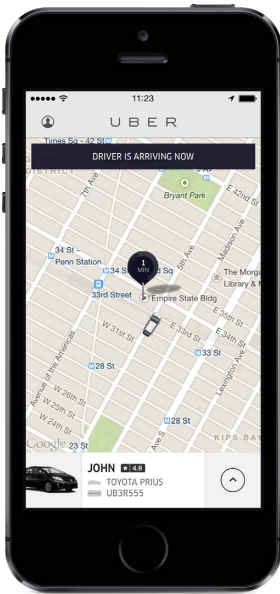
This is just as much about design as the energy data visualization that alerts you when you’re using more electricity than your neighbor.

In short, IoT gives us more and better data about the world. It gives us the ability to use that data to create smarter products and services that adapt to our needs and enable more efficient use of resources. The design challenges are in identifying opportunities to use data to create new products and services, and deliver better user experiences. As Tim O’Reilly points out, even Uber (Figure 13-4) is arguably an example of this:

The IoT is really about human augmentation. The applications are profoundly different when you have sensors and data driving the decision making... Uber is a company built around location awareness. An Uber driver is an augmented taxi driver, with real-time location awareness. An Uber passenger is an augmented passenger, who knows when the cab will show up. Uber is about eliminating slack time and worry. People would call it “IoT” if there was a driverless car, but it already is part of the IoT. You can measure, test, and change things dynamically. The IoT is about the interpolation of computer hardware and software into all sorts of things.<sup>5</sup>

---

5 [http://bits.blogs.nytimes.com/2015/02/04/tim-oreilly-explains-the-internet-of-things/?\\_r=1](http://bits.blogs.nytimes.com/2015/02/04/tim-oreilly-explains-the-internet-of-things/?_r=1)



**FIGURE 13-4**

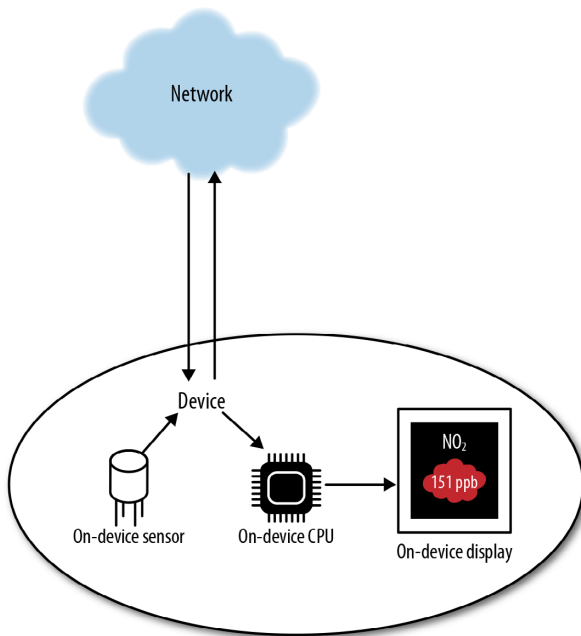
Uber's service relies on location data (image: Uber)

## FLOW OF DATA IN CONNECTED PRODUCTS

In this section, we'll look at the technical context around data in IoT systems: where the data comes from, where the data is processed, how frequently the device connects to the network, and what may be done with it. The traditional Web is very centralized, but in IoT the edge devices can be a lot more autonomous. In Chapter 3, we discussed the need to understand “where the code runs” in a distributed system and the impact that it has on UX. In a data-driven product, where the data comes from and the way it flows around the system are also fundamental to shaping the user experience.

We'll start by looking at this from the perspective of a single device (Figure 13-5). Data may come from onboard sensors on the device, or from the network. It may be processed via the device's onboard CPU, or in the cloud. The device may transmit data all the time or store and sync intermittently. Data may be displayed on a local UI on the device (or otherwise acted on locally), and/or passed to other devices, via local networking or the Internet.

A simple example might be a single air quality sensor out of a network of hundreds or thousands. Each individual sensor takes a data reading and sends it to the network at intervals of a few minutes. It may have enough intelligence to transmit more frequently if readings go over a certain threshold, and the ability to cache readings and send them in a batch if it loses connectivity. But it isn't doing any local processing to analyze or transform the data in any way. All the processing is done in the Internet service. The early version of the Jawbone UP (Figure 13-6) and the Withings scale also work on this model. The local device lacks a UI (or if one is present, it's minimal): the data is mainly accessible via a web or phone app.



**FIGURE 13-5**

Flow of data around connected devices

A more complex example might be a robot vacuum cleaner, which uses data from onboard sensors and onboard processing to navigate and learn the layout of rooms for more efficient operation. In normal operation, it doesn't rely on the network—all processing is local. It is not, for example, downloading floor plans of the house. But it might connect to an Internet service to report details of its activity (so the user can see when it has cleaned) or diagnostics (so the user can tell when it needs

servicing). A thermostat that attempts to learn when the house is occupied, and what temperature the occupants find comfortable, might also do a significant amount of local processing. In addition, it might also have an onboard UI.



**FIGURE 13-6**

The original Jawbone UP did not process data locally (image: Jawbone)

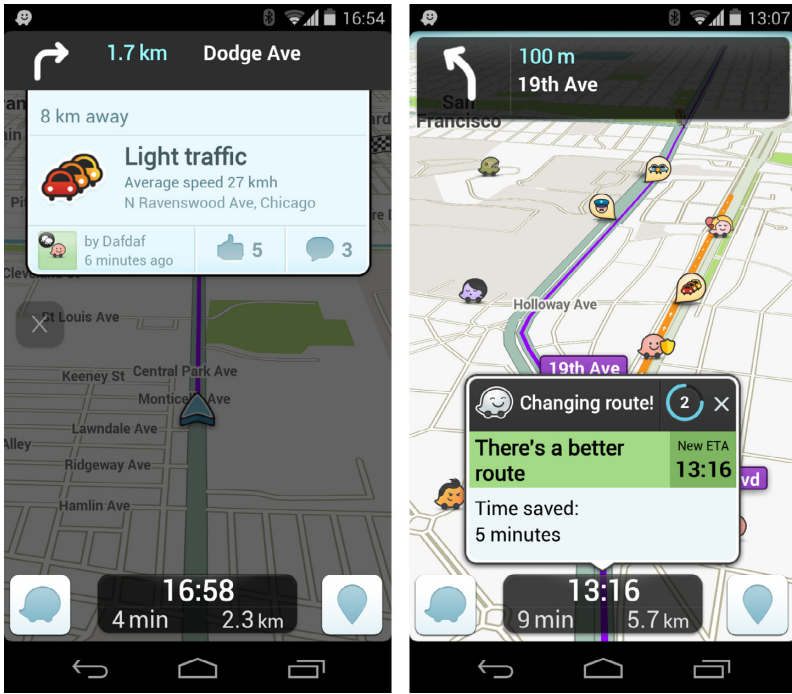
An example in between the two might be a car navigation system that uses a combination of networked data (live traffic), onboard readings (GPS), and static onboard data (maps). Some processing is applied locally (e.g., finding the best route to the user's intended destination given all available data), and some remotely (e.g., traffic information). Waze,<sup>6</sup> a car navigation app, crowdsources live traffic information from its users to determine the best route. In this example, the edge device broadcasts data to the network as well as receiving it, allowing the service to build up an aggregated picture of local traffic conditions across all users<sup>7</sup> (Figure 13-7).

There are pros and cons in handling data processing locally (in edge devices) versus centrally (in a cloud service). This is a key question that system architects need to consider, and the best answer for any given system will depend on the characteristics of that system.

---

<sup>6</sup> <https://www.waze.com>

<sup>7</sup> In the case of Waze, the edge device is a smartphone, but this pattern could easily apply to a connected device as well.



**FIGURE 13-7**

Waze crowdsources local traffic and road information in real time (images: Waze.com)

Cloud servers are more powerful than edge devices, can take in more data points, and more easily connect to other services to draw in data from other systems. Handling analysis in one place means that network traffic can usually be reduced, as the service pushes out the results of the data analysis (instead of a network of devices pushing raw data at one another). This kind of centralized processing is good for large datasets like energy analytics or trends in fitness data. It scales much better than local processing: as you have more data and more capacity, you can add more servers. Because storage is cheap and scalable, data in the cloud is also persistent: you can keep it as long as you need it. Data in an edge device may need to be erased when storage space is needed.

However, as we've seen in Chapter 3 and throughout this book, anything that happens over a network is subject to latency and reliability issues. If you need immediate responsiveness or near-failsafe reliability, a device will need to be able to process data locally to some extent—for example, a self-driving car equipped with cameras, image

processing, and proximity readings detects a potential collision situation. The potential course of action—swerve, slam on brakes, do nothing—may be different, depending on whether the object in the way is a wall, child, or squirrel and whether taking that action might endanger anyone else on the road. That decision must be taken and acted upon in a split second, which means local processing. The car cannot risk having to ask the Internet what to do.

Many current IoT services use a centralized model. But interest in processing at the edges is growing. Cisco uses the term “fog computing”<sup>8</sup> to refer to the distributed model of cloud computing that encompasses sensors, smarter edge devices, and smarter hubs as well as centralized servers. This kind of distributed processing may protect against latency and outages and enable better user experiences.<sup>9</sup>

## DATA SCIENCE

Whether you’re working directly with data scientists, or third-party data that was analyzed elsewhere, it’s worth having a basic understanding of what data scientists do.

Data science is the extraction of actionable knowledge from data.<sup>10</sup> Data scientists explore complex problems using a variety of techniques from many fields, including mathematics, statistics, and computer science.

A particular focus in current data science is big data. Extracting results from large, distributed, unstructured, or semi-structured datasets is a very different challenge from querying organized, structured databases. It has required the development of new types of algorithm (e.g., Google’s MapReduce). One of the central trade-offs in big data is in optimizing the data for fast querying versus leaving it in its original form. Preserving the original source provides the flexibility to use the data in other ways in the future, but carries the drawback that running queries will take more time.

---

8 <https://techradar.cisco.com/trends/Fog-Computing>

9 Alistair Croll, “Clouds, Edges, Fog, and the Pendulum of Distributed Computing,” in *Big Data Now: 2014 Edition: Current Perspectives from O’Reilly Media*, January 2015, page 46.

10 [http://en.wikipedia.org/wiki/Data\\_science](http://en.wikipedia.org/wiki/Data_science)

Accommodating and analyzing large volumes of data is a serious investment in both effort and money, and has to be well planned. Gathering any data you can in the hope it will be useful later is not efficient or effective.

As with any kind of science, starting with a clear formulation of the questions you want to answer with data is important (as well as the business objectives of spending time investigating it). The questions frame the data you choose to collect, and how you analyze them. How you analyze them, in turn, has an impact on your conclusions.

Cleaning and preparing data (e.g., dealing with errors and inconsistencies, weeding out data that doesn't help answer your questions) prior to analysis can take a significant amount of time.

Analysis involves extracting meaningful insights from the data, and can also be extremely complicated to get right. An activity tracker might use an accelerometer to capture movement. But patterns of movement vary depending on the activity type, your height, your weight, and even the type of shoes you wear. If your goal is to get an accurate picture of the user's activity level, you might require data from other sensors or sources. A smart energy metering service might try to disaggregate the consumption of each household appliance from a customer's energy readings.<sup>11</sup> Or it might compare a single customer's energy readings against a reference data set for similar households to produce similar insights by statistical inference (see Figure 13-8)

Finally, there is the communication process, which may involve visualizing the results, translating numeric insights into the story the data is actually telling, and the actions it points toward.

## **TYPES OF IOT DATA**

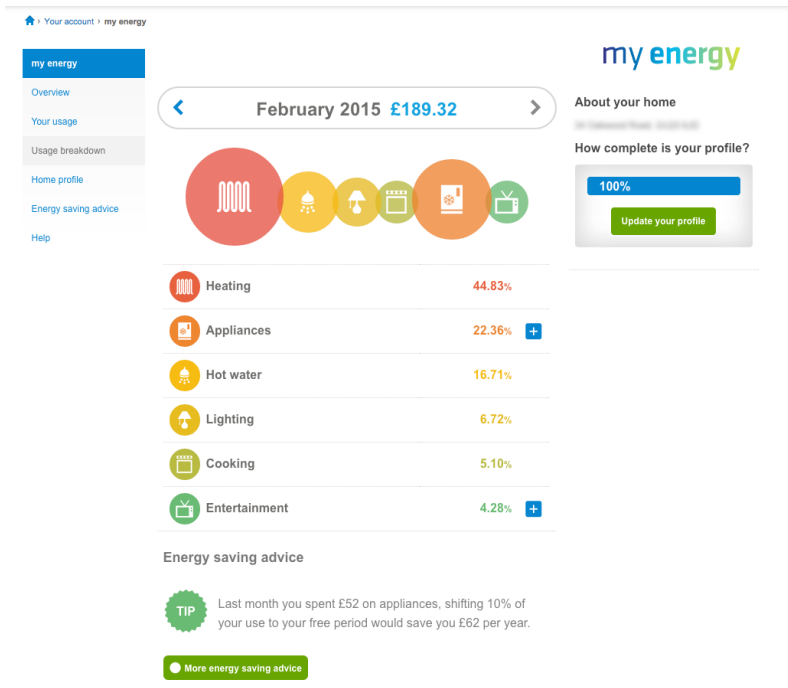
The types of data that may be captured by connected devices includes:

- Information about the physical world (e.g., it's 14° C, there is no movement in the living room, NO2 is within acceptable limits)
- Information about things, such as location or state (e.g., the package is in Jacksonville, the light is on and using 13W, the washing machine has a blocked pump)

---

<sup>11</sup> <http://bit.ly/1P3Ytj2>





**FIGURE 13-8**

The British Gas My Energy online report shows a breakdown of a smart meter customer’s energy consumption by appliance (image: British Gas)

- Biometrics (e.g., heart rate, respiration)
- Human behavioral data, such as physical activity level or what the user is doing with the system (e.g., turning on the oven, accelerating and braking in a car)

A system may well use a combination of all of these. For example, doing a run with a fitness tracker (or even just a smartphone) would generate data about the location, time, speed, and number of steps taken during the run. Wearing a heart rate monitor would add heart rate measurements, which could be used to infer the intensity of the run, in relation to the user’s general fitness level. The system might also track weather data from a third-party service, to take into account run conditions and the effect on the body.

### **Static versus dynamic**

Data can be either static or dynamic. Static data doesn't change, or changes very slowly. Static data about the world includes mapping data, such as the locations of countries, mountains, cities, and buildings. Static data about a person would usually include their sex and name, height, and date of birth. Static data about things tells us about their properties: for example, the washing machine has a maximum spin speed of 1,400 RPM, or the car is a red Ford Focus with a two-liter engine.

Dynamic data changes all the time (i.e., particular measurement, taken at different times, will produce different results). Dynamic data about the world includes temperature and humidity readings, weather conditions, traffic density, levels of pollutants, and whether there is currently movement in front of a sensor. Your weight, activity level, blood pressure, and energy intake are all dynamic. Dynamic data about things describes what they are doing right now (e.g., the washing machine is currently in rinse cycle, the fridge is using 45W of electricity, the boat is at latitude 17.761158 and longitude -88.027749).

In an IoT context, static data generally serves to provide interpretive context to dynamic system data, such as sensor readings or information about what the system (or user) is doing. For example, the GPS coordinate just mentioned is meaningless to most of us on its own. But set in the context of mapping data, it can be used to tell us that the boat is moored in Caye Caulker, Belize (see Figure 13-9).

### **Direct versus inferred**

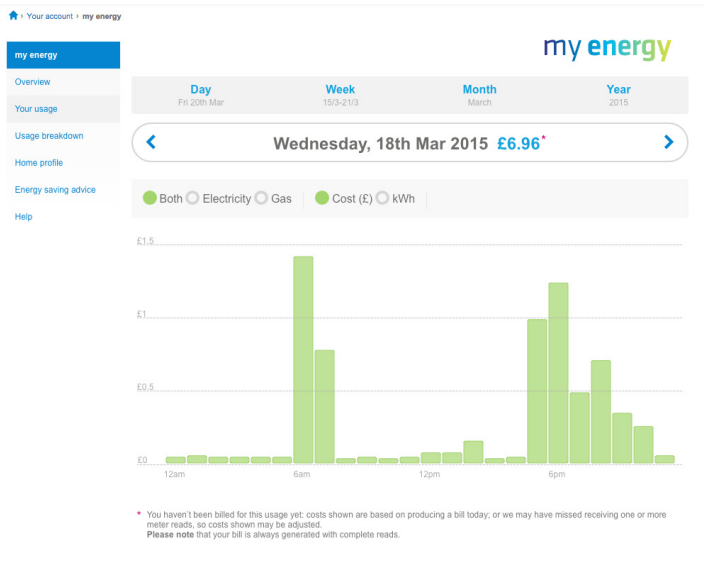
Some data points, such as the user's heart rate or current speed of driving from the odometer, are directly measured. There may be some question over the margin of error of the instrument used to measure them, but in general we treat such measurements as facts about the world.

Other observations may be indirectly inferred from other sources. Observing electricity consumption patterns in the home can help indicate whether anyone is in, up and about (see Figure 13-10). If a bike is traveling away from its owner's phone, it might be stolen. If bed sensors detect that a senior citizen is not up by 9 a.m. and has not boiled the kettle as per her normal routine, it could be an indication that something is wrong.



**FIGURE 13-9**

GPS coordinates 17.761158, -88.027749, aka Caye Caulker (images: Sam Beddoes via Flickr, CC licence; Google Maps)



**FIGURE 13-10**

Hourly energy consumption patterns for a day give an indication as to when the home's occupants are in, and active. (Image: British Gas My Energy online report)

Inferences can be extremely powerful, but they contain the potential for inaccuracy. If your system acts on inferences, or presents them to users, it's important not to treat them as incontrovertible facts.

### **Big versus small, real-time versus historical**

Data scientists often focus on “big” data: huge, distributed datasets and the techniques for working with these. As we've discussed, IoT will create *a lot* of data that needs to be processed and analyzed in this way.

But data does not have to be big to be significant. A single sensor reading can tell you that there is smoke in your home. A single GPS coordinate from a pet collar can tell you the location of your lost dog. Very small data, like this, can be highly significant in telling us what an object is doing, or what is happening in the environment. Small data points like this are generally most significant when they tell us what is happening right now. Knowing that your child has just come home from school or that your house is using 2kW of electricity is useful at the time, but much less so four hours later and not at all two weeks later. Occasionally, a small data point can remain significant in a historical context, if it is linked to a significant event. For example, knowing exactly what time the motion sensor triggered your alarm, and in what room, when you were burgled two months ago could be important information in support of an insurance claim or criminal case.

These data points mostly have significance in the longer term only when aggregated to identify trends, or provide meaning to current readings. Knowing that your child usually comes home at 4 p.m. or your house's baseload electricity consumption is 150W is helpful to provide context to things that happen now. For example, if you're on vacation but your house is consuming considerably more electricity than your baseload, something has been left on, and you might want to turn it off (if you have the means to do so).

Over time, individual data points are likely to decay in value more and more, and become useful only as part of aggregated trends. Your home's energy consumption day by day, last winter, might be useful mapped to weather conditions. It might give an indication as to how efficient your heating system is. Your home's energy consumption in January two years ago may only have value as a single aggregated figure, used to show that this January's consumption has been high, low, or average compared to previous years. The user's interest in the data is

likely to become less granular over time. It often makes sense for more recent data to be fine grained (e.g., to see every weight measurement you have taken recently). The older that data becomes, the more coarse-grained the focus of interest becomes: comparing your recent weight with general trends up and down in recent years.

Large datasets often contain historical data, but complex IoT systems can also generate a large amount of live data that needs to be analyzed and acted on in real time. Real-time data enables context-aware services that modify the information, functionality, or even pricing offered to the user in response to live conditions. Uber<sup>12</sup> monitors the location and availability of cars across the network to determine when demand is high, and charges more at those times: this is called surge pricing. Surge pricing can help to manage demand, by encouraging more drivers to work at busy times.<sup>13</sup>

### **Time, frequency**

When data is being aggregated from many different sources, keeping track of the order in which it happened (time series) is critical. Knowing whether a reading from sensor A was taken before a reading from sensor B, or vice versa, can completely change the interpretation of the data. Is the house warming up, or cooling down?

The frequency with which data readings are taken is also important. “Real-time” data often isn’t actually live. As we saw in Chapter 3, many connected devices aren’t constantly connected, so may report or receive data intermittently. Sending very frequent readings may clog up the network. “Live” data readings might be taken at intervals ranging from a few seconds to a few minutes or more. As cited in Chapter 4, an electricity smart meter sensor might report 7-second reads, but a gas sensor might only report 15- or 30-minute reads as it runs off a battery. This changes the value of the data. The electrical data can perhaps tell you which appliances are on right now that you might need to turn off, and the gas data will give you general trends, such as whether the

---

12 <https://www.uber.com>

13 Although there have been allegations that drivers are gaming the system: getting together in large numbers to go offline, reducing availability, then going back into service to benefit from raised prices. See [http://www.reddit.com/r/uberdrivers/comments/zdd171/driver\\_driven\\_surge\\_pricing/](http://www.reddit.com/r/uberdrivers/comments/zdd171/driver_driven_surge_pricing/).

heating schedule is aligned with the times the house is occupied or not. As we saw in Chapter 3, power issues are a major factor in determining connectivity patterns of connected devices.

### **AUGMENTING YOUR DATA WITH THIRD-PARTY DATA**

You may choose to augment the data from your product with third-party data. This is often used to provide context. For example, third-party weather data might help an energy monitoring system interpret where energy is being used, or enable more efficient functioning of a solar power generation system. A connected lost-and-found tag on your bag might allow you to link its service with your account on Uber and Foursquare, allowing you to backtrack to places where you might have misplaced your item.

If you're using data from a third party, you will have to work with their data structure, and you are at the mercy of their availability. In Chapter 3, we discussed rate limits on APIs, which might mean that you are only able to query the third party's data API a certain number of times per day, or per hour. You may not always get "live" data, especially not from a free third-party service.

If your product is sold in different markets, there may be geographic differences between the third-party services used between those markets. Public transit information for London does not come from the same source as public transit information for New York.

Integration with third-party services may require explicit consent from the user, for example, if you want to access their Facebook friends graph. You need to design the service to degrade gracefully if the user refuses or revokes permission, or if the third-party dataset is temporarily unavailable.

## **Types of Data-Driven Product**

Designing with data isn't just about presenting data to users. As we saw in Chapter 4, data gives us the potential to think about products differently.

The current crop of products on the market can be thought of in terms of:

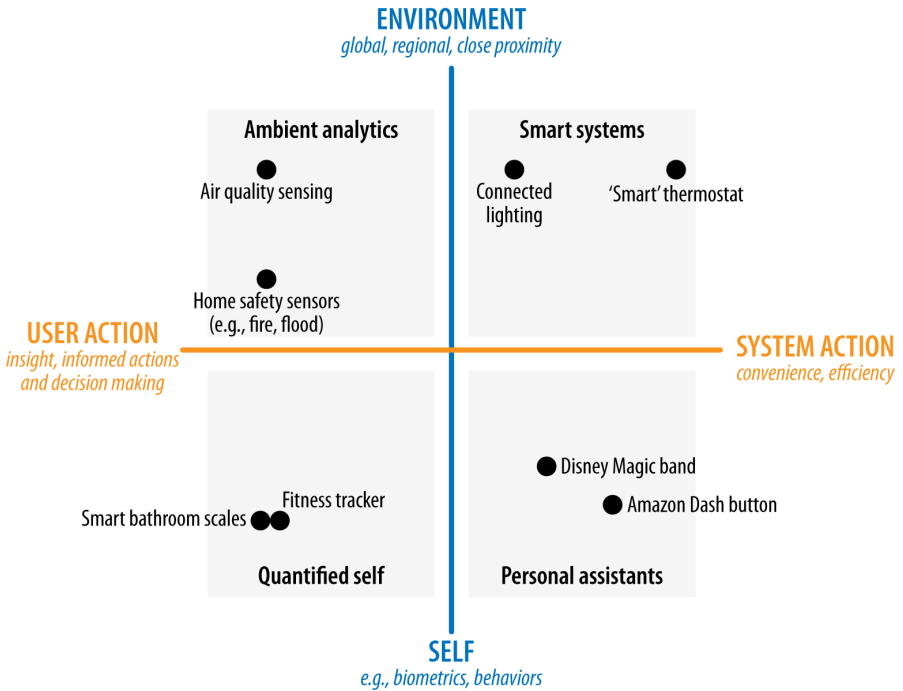
### Context

How the data relates to the user: whether it relates to them as a person, or their environment

### Action/automation

What is done with the data: whether it is presented back to the user to inform his own actions, or whether it's used to drive system actions, like automation or autonomous systems

The grid in Figure 13-11 shows how this translates to four product quadrants.



**FIGURE 13-11**

The context-action framework

The vertical axis represents the context of the data in relationship to the consumer. At the top end of the vertical axis are broad-scale data about the environment, such as atmospheric temperature or traffic. Further down, we might find more localized information about the user's personal environment, such as which light bulbs in her home are switched on and how much energy the home is using right now.

At the bottom of the vertical axis is data about the users themselves. These could be physical measurements such as heart rate, glucose level, diet, and energy consumption through daily activities, or behavioral data such as sleep patterns or how much time they spend using a smartphone.

The horizontal axis represents what that data is used for.

On the left is user action, where data is presented to the user, who then decides what action to take. Many current IoT products fall into this category. The Automatic Link (see Figure 13-21)<sup>14</sup> is an example of an insight product. It aggregates sensor data inside the car with gas prices to help consumers relate fuel cost to their driving habits. It doesn't do the driving on their behalf.

On the right is automation, where devices act upon the data themselves. This may be through automation—where the user creates rules to be triggered when certain data values are met (e.g., turning the lights on when it gets dark, or unlocking the car when the user approaches). Or it might be through systems that take autonomous action (e.g., a thermostat that learns when the home is likely to be occupied and what temperature the residents find comfortable). The goal is to save the user effort, and sometimes to optimize/produce a better result than could be achieved by letting the user do all the work (i.e., setting heating schedule). (Automation and autonomous systems are covered in more detail in Chapter 15.)

Over the next few years, we can expect to see some product categories move from insight toward automation. For example, smart meters currently only offer insights into energy usage. The emergence of the smart grid—able to predict outages and balance demand—will enable energy suppliers to offer dynamic pricing and incentives to customers willing to allow the supplier to control their usage (e.g., pausing the tumble dryer) at peak times. But there are some activities that won't lend themselves to automation. An activity tracker, like the Jawbone UP or Fitbit, can tell you whether you're getting enough exercise, but will never be able to go for a run on your behalf (although perhaps in the future it might offer a tie-in to your diet plan or grocery shopping service to help you eat more healthfully).

---

<sup>14</sup> <https://www.automatic.com>



Each quadrant of this diagram represents a different type of product in the consumer IoT market today. Next, we'll look at what those are, and what that means for design.

## SMART SYSTEMS

Smart systems aim to achieve efficiency through automating actions. Most of these systems are comprised of multiple devices distributed in the environment, such as home automation systems. Key design challenges are interusability and interoperability. The ability to work with other systems in the same environment (e.g., the security alarm and the lighting system) magnifies its value.

## AMBIENT ANALYTICS

Ambient analytics systems (Figure 13-12) seek to improve the consumer's understanding of their environment. Drawing on data from sensors and (often) external sources, they seek to expose trends and changes in an environment. The goal is to provide reassurance, and when appropriate, enable the consumer to proactively make decisions against anomalies.



**FIGURE 13-12**

The Netatmo weather station can track temperature, humidity, air quality, and indoor and outdoor carbon dioxide level; the consumer can view data trends using an online dashboard, and share data with others around the world (photo: Masaki Okomura/Netatmo.com)

## PERSONAL ASSISTANTS

Personal assistants (Figure 13-13) use the consumer's data to personalize services and remove pain points from their activities. Because these solutions are expected to be personal, the system can deliver additional value by learning and adapting to changes in the consumer's patterns. As with smart system solutions, interoperability with other systems is often important: for example, the Disney MagicBand works with payment systems, hotel room access, restaurant and ride bookings, and more. Safeguarding the consumer's personal data is vital for these systems. Amazon's Dash Button provides a trigger allowing the user to reorder branded consumables at the point at which they are used. The Dash Replenishment Service behind the button relies on Amazon's ordering, payment, and fulfillment systems and can be built into the device hardware by the manufacturer. Any unauthorized change to the data may trigger false actions in different parts of the system.

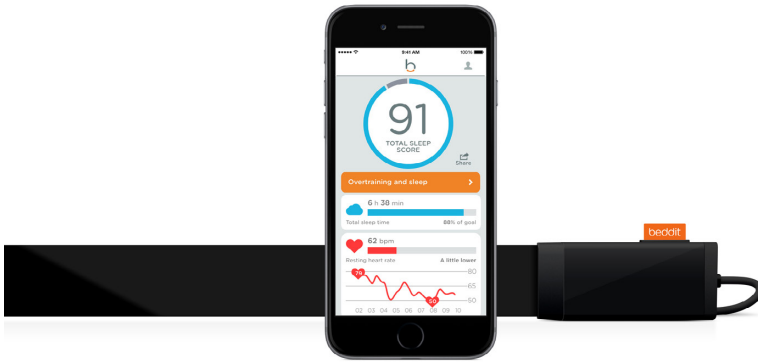


**FIGURE 13-13**

An Amazon Dash button for ordering Tide laundry detergent on a washing machine (image: Amazon.com)

## QUANTIFIED SELF

Quantified self products (Figure 13-14) help the consumer understand their behaviors and provide insights to help them make improvements. Most activity and sports performance trackers are in this quadrant. The majority of the products in this quadrant are bought with a specific goal in mind, often linked to behavioral change. The key design challenge is to use the insights to help users toward that goal. This requires effective feedback grounded in behavioral psychology.



**FIGURE 13-14**

The Beddit Sleep Monitor: a thin sensor strip adheres to a mattress and tracks the consumer’s sleeping quality based on his movements, respiratory rate and heart rate; the accompanying mobile app visualizes the data and provides recommendations to improve the consumer’s sleeping habits (image: Beddit)

Many quantified self products are well-being oriented, targeting those who are generally well but wish to be healthier or fitter. More products are likely to emerge that target those coping with ill health, or the risk of it. But medical devices must pass stringent certification processes with regulatory bodies like the FDA (in the United States). The manufacturer is also liable should any harm come to the user through a fault in the device, or inaccuracy in the information. As a result, many companies find it easier to frame their connected devices as “lifestyle” products rather than medical-grade equipment.

### **DATA ABOVE THE LEVEL OF A SINGLE PRODUCT**

This last example highlights that the value of data-driven products isn’t just about the individual user. The more connected products we have, the greater the value of the aggregated data they generate. Data from a single car may be useful to the driver, but data about many cars provides a view of traffic density. Data from a single activity tracker and heart rate monitor might help the watch’s owner manage her fitness, but data from thousands or millions of similar devices, anonymized but tied to basic demographic information, could be an invaluable resource to medical researchers (or insurance companies).

For example, Apple's ResearchKit—a platform enabling medical researchers to collect health information from iPhone users—has just been announced at the time of writing (though not yet released). It's easy to see how heart rate data from the Apple Watch across millions of wearers could be mined for cardiology research.

## What This Means for Design

In this section, we'll look at the key considerations for creating valuable user experiences for data-driven products.

### **MAKING MEANING AND ENABLING ACTION**

The most fundamental questions when designing data UXes are:

- How to make the data meaningful to the user
- How to make data actionable: enabling something of value to happen as a result of the data.

People often equate making data meaningful with visualization. Visualization can be a powerful tool for communicating meaning, but it isn't the whole picture. In this section, we'll consider general principles for identifying what's meaningful and valuable to your users, and how to present it to them.

The process of generating meaningful and actionable insights starts right at the beginning of the process of working with data. It affects the framing of the questions you want the data to answer, the choice of what data to collect or measure, and the algorithms you use to analyze it. As a designer, you may be involved in these data science activities, or you may be tasked with communicating the outputs to the user.

Either way, the key question (as ever, in user-centered design) is understanding the user's motivations and goals for using the product. Why are they using your product, and what do they want to achieve through using it?

Generally, users have an end goal in mind that has nothing to do with the data. That goal might be to lose weight, or save money on energy. By understanding what motivates your users, and what they want to achieve with your product, you can focus your data insights on helping them achieve that goal.

## Understand motivations and goals

Different underlying motivations and goals for using your product can put a radically different spin on the resulting UX. Imagine you're designing a home energy monitoring service. You might determine that there are three possible motivations that drive energy consumption behavior:

- Maintaining a reasonable standard of comfort and convenience
- Saving money
- Minimizing the impact on the environment

Different users may care about these things to different extents. Those who are time poor but cash rich will probably value comfort and convenience quite highly. Those for whom money is tight will be very keen to reduce their bill. And those who are very environmentally conscious will want to minimize their carbon footprint.

It's quite possible for users to care about two or all of these. But each energy decision they make (whether to switch on the AC or heating, whether to invest in home insulation or solar PV panels, or even just whether to leave on the landing light at night) is driven by a consideration of which is most important there and then. It's quite possible to care about minimizing your carbon footprint but be willing to leave a light on all night if an elderly guest prone to falls is staying. And however much you want to save money, you might draw the line at taking cold showers in the winter.

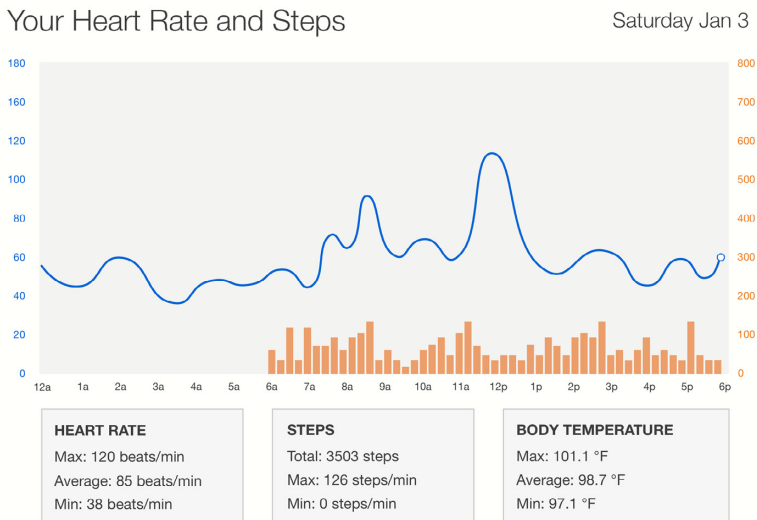
The ideal design for each of these motivations in isolation would be quite different. A cost-saving experience would break down in detail the cost implication of every energy behavior, maintain a running prediction of the next bill, suggest any possible savings, however small or potentially inconvenient, and keep track of the cost savings actually made. An environmental impact reduction experience might frame energy behaviors in terms of carbon impact, suggest ways to minimize it, track carbon saved from behavioral changes over time. A comfort and convenience-centered experience might suggest ways to reduce waste and raise comfort, such as investing in a new boiler or better insulating the home. It might offer automation—for example, via an intelligent thermostat. But it would not bug the user to make small savings all the time, especially if they involved effort or inconvenience.

In many cases, you'll be designing for a user base that has a mix of goals and motivations, and creating a balanced design that keeps everyone happy will be a big part of the challenge. In the case of the energy example, you might find that the common ground is reducing waste. Even people with pots of money don't generally like wasting it. Or it might be informed control: everyone wants to know the energy impact of their activities so that they can make informed decisions about where their money, or carbon usage, is going.

### Provide context

At first, just having data that wasn't previously available can be a novelty. Seeing your step count for the day or your home's electricity consumption pattern over 24 hours for the first time can be pretty interesting.

Suppose you wear a heart rate monitor/step tracker for a day. Perhaps you bought it because you want to monitor your overall health, or to get fitter, or get better sleep. The service sends you a graph of your daily readings in an email (Figure 13-15).



**FIGURE 13-15**

Heart rate and activity chart for a day

There are some visible patterns in this data. Your heart rate:

- Rose in the morning just after you woke up
- Varied throughout the day, correlated to your activity level
- Rose during lunch (around 12 p.m.), then fell and returned to normal about an hour later

In this chart, the data has been collected and synthesized into a readable form. It's interesting, the first time you see it. But what is it telling you? Are you normal? Are you healthy? Is there anything here you ought to be worried about?

In order to understand what the data means, you need more context. That might include comparisons to your previous data: how these readings compare to your heart rate in general (is anything unusual happening today?); how they compare to the healthy range for someone like you (are you generally fit?); how they relate to the reason you bought the product (improving your fitness or getting better sleep). To have value, insights need to be relevant and useful: helping users achieve their goals.

Providing context might also mean relating the data to the user's prior knowledge or experience. Part of this is about representing it in terms that make sense to people. For example, energy consumption is generally measured in watt or kilowatt hours (kWh). Most people have no idea what a kilowatt hour is. It's a pretty complicated thing to explain, not least because live electrical power consumption is measured in watts or kilowatts, which are similar but not the same. Some energy meter displays mix both watts/kilowatts and watt hours/kilowatt hours.<sup>15</sup> Most consumers have zero interest in understanding this. Comparisons can make the mystical energy units more palatable: you used 20% more energy heating your home this winter compared to last winter; your new TV's power consumption is 30% higher than your old one.

---

<sup>15</sup> A kilowatt-hour is a unit of energy equivalent to one kilowatt (1 kW) of power expended for one hour. A live power reading from a fridge might show that it's consuming 35W, or 0.035kW. The total energy consumption of the fridge over one hour would be 0.035kWh. To make things even more complicated, a dual fuel gas and electricity live display might show electrical power consumption in watts and gas energy consumption in watt hours. For the safety reasons discussed in Chapter 4, the gas reading is probably only taken every 15 minutes, so it's based on cumulative consumption not a live read.

From our own research, consumers generally measure energy consumption in terms of money. *How much was last month's bill? How much will the next one be?* A valuable way to help them understand their energy consumption is to relate their energy behaviors to cost. How much does it cost them to run the tumble dryer, have a shower, or to heat the house for a week? How much do the appliances that run all the time (their baseload consumption) cost them per minute? The British Gas report back in Figure 13-8 shows consumption by appliance in both cost and kWh.<sup>16</sup>

The precision with which you express your data also sets context. An activity tracker that indicates you walked 7,518 steps today implies that it is extremely accurate. As we saw earlier, this may not necessarily be the case. If what your system is good at measuring is general trends, focus the UX on highlighting those trends instead (e.g., you have been less active in the last week).

### **Actionable insights, explanations, and data**

If you have lots of data, it's tempting to want to show it off. But less is usually more to avoid the key insights becoming lost in the morass. Graphs and other visualizations can be powerful tools, but unfocused "dashboards" that summarize all the data to the user can be overwhelming. Designers and data scientists may love to look at beautiful visualizations. But mainstream consumers may associate graphs with math lessons in school. Not everyone enjoyed math.

Think back to the consumer's goal: is it to spend time looking at graphs, or is it to achieve something outside the product?

If you're using data to provide a smarter service, as is the case with the Disney MagicBand, you may not have a "data" consumer interface to speak of. The power of your service is in your ability to improve service interactions in context. Waiters can locate customers in the restaurant with the food they ordered earlier, and the service can plan itineraries

---

16 Representing energy usage in money may seem obvious but it isn't necessarily easy. Utilities often don't charge a flat rate for all kWh. They may have Byzantine tariff structures in which some kWh are charged at one rate and others at different rates. Providing an accurate cost for an activity may be near impossible. But an approximation of cost may still be more meaningful to the average energy consumer than an accurate kWh figure.

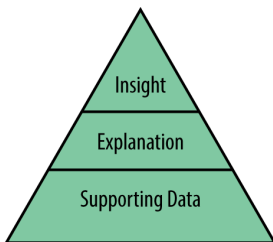


around the park attractions that minimize walking and waiting time. The key here is to improve the users' experiences at the "moments of truth" that most affect their overall experience.

If you are displaying data to users in a UI, your UX should focus on the headline insights. There may only be one or two of those, and that's OK. It might even be better to forget the graphs and just present text headlines—for example, "You're getting fitter. Congratulations!"

The headline should contain enough information that the user can see the relevance and grasp the important information without needing to read further. It should be specific, and where possible, provide some explanation for the insight (e.g., "Your next gas bill is on track to be \$100 more than usual, because the weather is colder than usual for this time of year").

You can then offer users who are interested access to a more detailed explanation of the insight, and then perhaps allow them to dig deeper into the data if they wish. This is the design technique of "progressive disclosure": giving the user only the essential information they need to understand at each point to reduce cognitive workload, with the option to dig deeper (see Figure 13-16).

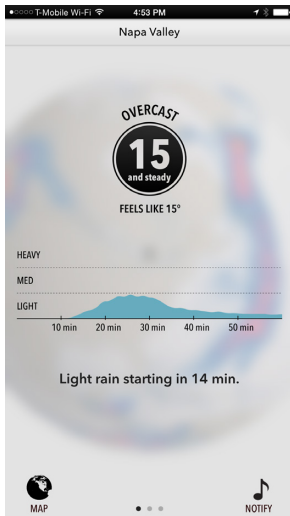


**FIGURE 13-16**

The pyramid approach to progressive disclosure: top-level insights are supported by explanations, backed up with supporting data for those that want it

The most important and valuable headline insights are those that are actionable in terms of users' goals. Can users do something as a result of this insight that helps them toward accomplishing their goals?

For a non-IoT example, consider weather apps. Conventional weather apps present facts about the forecast for your general area. The Dark Sky app uses hyper-local information to answer the question most likely to influence our short-term behavior: is it about to rain where I am now? (See Figure 13-17.)



**FIGURE 13-17**

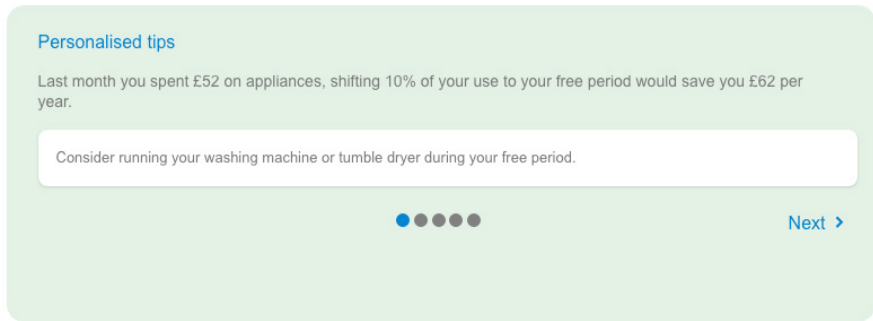
Actionable advice in Dark Sky, a mobile weather forecast app

The personalized energy tip in Figure 13-18 provides information on appliance usage linked to a suggested action to reduce the user’s bill.

An activity tracker might consider contextual data, such as weather and the user’s calendar, to offer actionable advice. Perhaps the user is running low on their target step count for the week, but it’s a nice day, they have nothing in their calendar around 12 p.m., so the system might suggest a lunchtime walk. This kind of advice is often far more valuable than a graph.

By enabling improved ways to control devices (whether manually and remotely or automatically), IoT gives us better opportunities to act on insights. The action might even be taken by the system—for example, the heating system that turns down the thermostat because it detects that the house is unoccupied and knows that the user has a broad goal of wasting less energy. Even without an “intelligent” heating system,

a service that correlated data from the security alarm and a connected heating controller could tell the user when the house was empty and the heating on. It could then give them the option to turn it off.



**FIGURE 13-18**

Personalized energy tip from the British Gas My Energy Report

Insights like these, which are actionable in the moment, require live or near live data. You can't offer an immediate action on out-of-date data. What "live" means will depend on your application. Electrical power consumption might be out of date in seconds—a room temperature reading can be minutes old and still assumed to be fairly accurate.

Live electricity data can tell you which appliances are on right now, allowing you to turn off any that are not needed. But if your gas data is 15 or 30 minutes old, it isn't necessarily telling you what is happening right now, but giving you an indication of daily fluctuations in energy consumption from which you can draw broader-brush conclusions, such as that the heating tends to come on before anyone gets home. In this case, the system might suggest you adjust the heating schedule, or offer to do it for you. But insights on longer-term trends may not lend themselves to such simple actions. Your gradually increasing weight over the last year can't be fixed with a button tap.

You may sometimes need to decide whether (and how) to communicate insights about things that the user can't act upon, or can't reliably control. A system that monitors the vital signs of a cardiac patient might calculate, based on data from other patients, that he has a 25% chance of having a heart attack in the next 48 hours. Should it tell him? A stressful and distressing message—"You might die tomorrow"—would be

counterproductive, not to mention unethical. A better message might be “Your heart is under stress. Get plenty of rest, take your medication and see your doctor as soon as you can.”

The timing and sensitivity with which insights are delivered is a major factor in determining their efficacy, and the user’s experience of receiving them. We’ll cover this in more detail in the following section.

## ATTENTION IS A SCARCE RESOURCE

Many of us already suffer from information overload. Connected devices will, if we are not careful, only make matters worse. The amount of information from devices on us and around us will dramatically increase, all competing for our attention. But human attention is a finite resource. Most of us are already interrupted many times a day by various types of message and notification. Interruptions are known to have a negative effect on our cognitive performance,<sup>17,18</sup> and are linked to stress.<sup>19</sup> Ubicomp pioneer Mark Weiser’s dream of calm computing seems an impossible utopia:

Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods.<sup>20</sup>

Current IoT systems like to tell us what they are up to, through smartphone notifications, email reports, tweets, SMSes, and even phone calls. That’s understandable while they are relatively novel. But the more of these devices we have in our lives, the more unmanageable the notifications and interruptions will become. Current methods of notification management require us to manage settings for each service on our smartphones, or within the service itself. That may not scale up

---

17 T. Gillie, and D. E. Broadbent, “What Makes Interruptions Disruptive? A Study of Length, Similarity, and Complexity,” *Psychological Research Policy* 50 (1989), 243–250.

18 J. L. Franke, J. J. Daniels, and D. C. McFarlane, “Recovering Context After Interruption,” *Proceedings of the 24th Annual Meeting of the Cognitive Science Society (CogSci 2002)*, 310–315.

19 B. P. Bailey, J. A. Konstan, and J. V. Carlis, “The Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface,” *Proceedings of the International Conference on Human–Computer Interaction (INTERACT 2001)*, 593–601.

20 M. Weiser, “The computer for the 21st century” *Scientific American*, Vol. 265, No. 3. (September 1991), pp. 66–75, <https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf>.

very far. We need ways to filter out the noise and stay informed of only what we need to know, which we can apply across multiple products and services.

We don't yet have good tools to do this. But as designers of IoT systems, we should still do our best to shape our services to be respectful of the user's attention, and consider carefully what justifies claiming it. We should aim not to add unnecessarily to the growing clamor of Internet noise that grinds all of us down at least sometimes. It is nearly always the case that users have something they view as more interesting or important to do than whatever it is we want to tell them right now.

We can do this by considering how the user's motivations shape:

- The relative importance of our messages
- The right way, and right time to deliver them
- How our services can help users filter information to get only what they need and want

The user's core motivations for using the product or service should help you focus only on the most important messages. Tell people only what they really need to know. Most people using activity trackers probably have the goal of trying to be more active. Instead of telling users how much they have walked, why don't these just alert users when they haven't walked enough?

The motivations for using a system should also shape communication patterns. For example, a core motivation for investing in security and many home control systems is peace of mind. A peace of mind system need not tell the user everything that is happening; that would be counterproductive! The user may occasionally want actively to seek reassurance: for example, did they remember to lock the door? But mostly, they want a system that runs quietly in the background when everything is OK, but that they can trust to tell them when there is a problem. The system needs to be smart enough to figure out when an event is urgent, important, or just noise, and whether it requires action. (We discussed the choice of notification channel in Chapter 12.)

Delivering messages at the right time is also key to being respectful of the user's attention. No one wants to be woken up in the middle of the night to be told their new sleep quality report is available. Understanding how your product fits into the daily or seasonal activities

of your users is valuable here. Are there natural cycles to using the product? For example, users may be interested in a live energy read at night to see whether they have turned everything off. Energy customers on a monthly payment plan are likely to be most interested in usage patterns on a monthly cycle, when the bill is due. Customers on a quarterly payment plan may take a longer view. Designing around these cycles may create a more meaningful product than assuming that they must want to look at it all the time.

Finally, consider whether your product can actively help manage information overload. This is a key opportunity for wearables. For example, a smart watch's configurable rules might allow users to filter smartphone notifications: being alerted only of the most important calls, texts, or notifications from the most important apps.

### **EXPERIENCE OVER TIME**

A key design challenge for data-driven products is what to show the user when you don't have very much data. The early experience of using a data product may, frankly, not be that great. It might need time to gather some data before it can do what it's really good at. The energy meter that only has one week's worth of data might be able to estimate that your kettle costs \$.04 to boil but can't tell you much about your long-term usage trends or predicted next bill. Engaging the user, and keeping them engaged, while you wait for the really valuable insights or functionality to emerge, is a challenge. User expectations need to be clearly set, and progress toward the more valuable stage of operation clearly communicated.

For example, the Nest Learning Thermostat makes its purpose very clear to the consumer up front—it keeps your home comfortable and saves energy by learning about you. During the first week when there is no data about the consumer's home, it asks the consumer to adjust the temperature of their home manually (this may even be a step backward in user experience to users accustomed to programmable controllers). Every time the user makes an adjustment, the thermostat provides visual feedback that it has learned the input. After a week of monitoring the temperature inside the home, adjustment patterns and movement patterns, it offers the user the option to turn on the Auto-Schedule. As the name suggests, Auto-Schedule tunes the temperature of the home

according to the user's previous adjustments. The act of asking the user to turn on the feature, together with the data that the feature provides, demonstrates that the device has been learning.

## **INFERENCE AND INTERPRETATION**

In “Types of IoT Data,” we discussed inferences: taking a set of observations, and applying inductive reasoning to conclude what may be happening, and why. If smoke is detected and the home's temperature is unusually high, there might be a fire.

Inferences can be very powerful, but they contain the potential for inaccuracy. If your product makes inferences about users, it's really important that they understand how you are forming those inferences, what you have done with them, and if they aren't accurate or they don't like the result, what they can do about it. We don't know whether the “smart” drink vending machine in Chapter 11 will tell you the age and sex it thinks you are and the decisions it has made to show you only certain drinks on that basis. But at the very least it ought to provide an option to view all drinks, to everyone.

A related problem is the interpretation that may be applied when trying to put your data in context. For example, an energy monitoring system might tell you that you use 50% more energy than “similar” households. You may well want to see how it has classified those households as similar (e.g., location, age and size of building, and number of occupants). If any of that information is incorrect, you'll want to update it (see Figure 13-19).

This is particularly sensitive when the product is perceived to be judging people for doing something “bad” or trying to modify their behavior. Users may feel they have valid reasons for their actions that the system isn't taking into account. You may genuinely be missing some context. For example, high energy consumption might be due to a sick person at home who needs the heating on all day. Sudden braking might not be bad driving, but avoiding a child running across the road. Poor sleep might be due to a new baby (see Figure 13-20). Be sensitive to the fact that your assumed view of the user's needs might be incomplete, and adjust messaging accordingly. The sleep app, for example, might communicate a change in data by saying “I've noticed a dramatic change in your sleep habits,” and offering to turn off alerts temporarily alongside guidance on getting better sleep.

100% complete my energy

About your home ✓

House type

Flat
  Detached house
  Semi-detached house
  Mid-terrace house
  End of terrace house
  Bungalow

House built range

Before 1919
  1919-1944
  1945-1964
  1965-1980
  1981-1990
  After 1990

Number of bedrooms

Studio
  1
  2
  3
  4
  5+

How many live in your home

1
  2
  3
  4
  5
  6+

Energy efficiency rating

A
  B
  C
  D
  E
  F
  G

Wall insulation
  Loft insulation
  Glazing

Loft insulation thickness in millimeters

None/Don't know
  50-99
  100-199
  200+

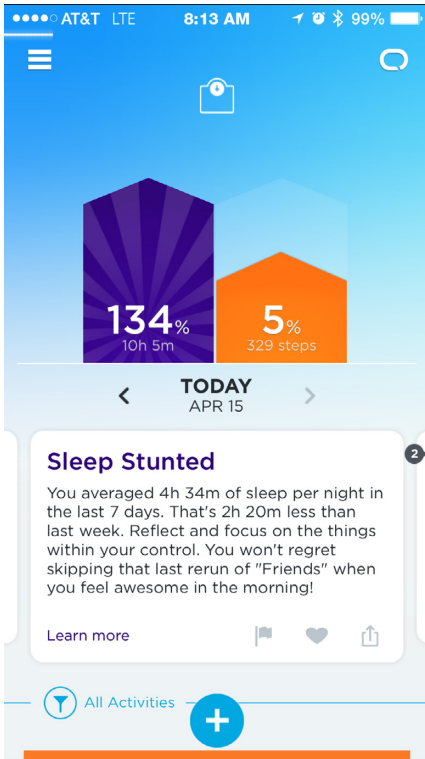
Percentage of double or triple glazing

None
  Quarter
  Half
  Three quarters
  All

**FIGURE 13-19**

The British Gas My Energy report allows users to provide contextual information about their home to aid more accurate interpretation of energy data (image: British Gas)





**FIGURE 13-20**

The Jawbone app showing evidence of poor sleep; this might be due to late night TV...or a sick child

## TRANSPARENCY AND TRUST

Where data about users is being collected and stored, the security of that information and the privacy of the end user are vitally important concerns. This is discussed in depth in Chapter 11.

In terms of UX design, the basic principles are:

- Be transparent about what data is collected and why, and don't collect more than you need
- Be transparent about what is done with it: how do you use it, and with whom it might be shared

Many products bury this kind of information within user agreements that almost no one reads. In 2013, LG smart TVs were found to be transmitting information about user's viewing habits and even the names of

personal files on USB sticks back to LG, for use as part of their Smart Ad targeting service, capable of offering “sharp suits to men, or alluring cosmetics and fragrances to women.”<sup>21</sup> LG claimed that the existence of this functionality was made clear in the terms and conditions and that it was the users’ fault for not reading these, but this argument was undermined by the fact that option to turn this functionality off did not work (this has since been fixed).

A much better approach is to communicate the benefits of granting the service access to the customer’s data. Show that the customer needs to grant access to a basic set of data in order to use the product, and adding additional data afterward brings additional features.

Also, make the data that you collect from your customers available to be downloaded as a report or raw data. Not everyone will use this feature, but the point is to make the information available for the customers who want to see what you can see. If any part of your service experiences a security breach, you must report the incident to your customers and communicate the steps that you will take to fix the situation. Develop a procedure to handle legal information requests from government bodies, national or overseas, to provide notice to users when such requests occur.

If a customer decides to stop sharing their data with you or to stop using your service, they should be given the ability to remove their data from your service.

## **DATA MAY NOT BE OBJECTIVE**

It’s important not to place too much faith in measurements and models. Data may give the appearance of representing immutable and precise truths about the world, but this is misleading. The way you take measurements, and the algorithms you use to analyze them, can result in different conclusions. Whatever you do, your data will only ever represent a partial worldview and it might not be as accurate as it appears to be. For example, your activity tracker may report that you took 7,518 steps today, but different brands of tracker can give different readings depending on variations in sensors, algorithms, where they are worn

---

21 <http://doctorbeet.blogspot.co.uk/2013/11/lg-smart-tvs-logging-usb-filenames-and.html>

on the body, and the height and gait of the wearer.<sup>22</sup> Algorithms can also learn the inherent biases and discrimination in the data and societal context, leading apparently objective analysis to produce distorted and discriminatory conclusions. For example, St. George’s Medical School created a big data algorithm to increase the objectivity of its admissions process. But the algorithm learned to favor white male candidates, as they were over-represented in historical admissions data.<sup>23</sup>

The sheer volume of data available now also has its risks. Humans are wired to perceive patterns in data, even if none exist. And if you analyze a large enough dataset, you will find random correlations that aren’t significant. A conclusion that appears to be supported by data will appear more credible, but it may not always be accurate.

The upshot for UX is always to be up front about what you measured, what you did with it, and its limitations. Make the data and the process as transparent as possible, and provide access to the raw data so others can, if they wish, explore it for themselves.

## **WHEN BEHAVIORAL CHANGE IS A GOAL**

There is a current trend for connected products that nudge us toward behavioral change: using less energy (e.g., Opower), getting more exercise (e.g., Jawbone UP), driving more fuel efficiently (e.g., Automatic; see Figure 13-21). Connected devices can be effective in supporting for behavioral change as they provide the ability to monitor behaviors and deliver insights and nudges in the moment. To be effective, such products require careful design and a good understanding of your users, the domain, and the relevant psychology around behavioral change. This is a deep and complex topic in its own right. If you’re interested in exploring it in depth, we recommend Stephen Wendel’s book *Designing for Behavior Change*.<sup>24</sup> In this section, we will give a brief introduction to some key concepts and examples.

Many of these systems track user behavior, such as activity level, and then provide insights on that behavior, nudging users toward “better” behavior that helps them toward their goal, such as walking more.

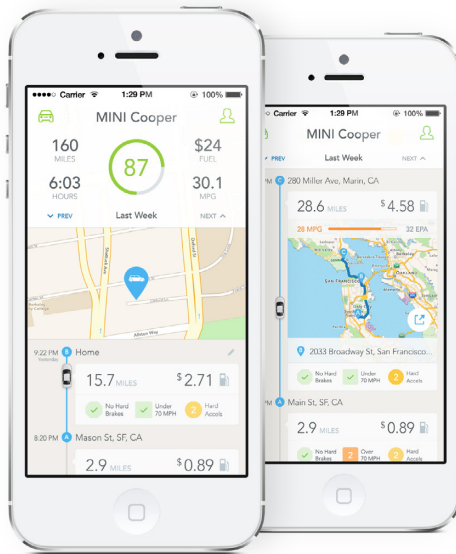
---

22 <http://www.bbc.co.uk/news/technology-31113602>

23 <http://www.thebolditalic.com/articles/4502-can-big-data-be-racist>

24 Stephen Wendel, *Designing for Behavior Change* (O’Reilly, 2014).

Sometimes it can be effective to take the behavior out of the user's hands, allowing intelligent, automatic technologies to make "better" decisions for us, like turning the heating off when no one's in. In this section, we'll focus primarily on the former.



**FIGURE 13-21**

Automatic seeks to help drivers learn fuel-saving driving habits. The Link device (left) relays data from the car's onboard sensors to the user's smartphone. By analyzing the data, Automatic visualizes the correlation between driving behavior and fuel consumption and gives the user a 'drive score' every day. Users can compare their scores with other users and are encouraged to minimize harsh acceleration, braking and control their speed for better fuel consumption (images: Automatic.com).

A central concept in behavioral change is the feedback loop, a model often used by behavioral scientists and service designers to help people create new habits.<sup>25</sup> There are different versions of the feedback loop, but in general there are four stages:

#### *Evidence*

Data about the behavior is measured, captured, and stored

#### *Relevance*

Evidence about the behavior is crafted into meaningful insights for the user

#### *Consequence*

The evidence is used to describe the outcome and effects of the user's current behavior

#### *Action*

Alternative courses of action are set out, encouraging the user to choose to change course

For example, an energy monitoring service might tell the user: “Based on analysis of your gas usage (*evidence*), we think you spend 20% more on heating your home than other similar households (*relevance*). This winter, that could be a difference of \$150 on your bill (*consequence*). Turning the heating down by 3° F could save you \$75 (*action*).”

Notice that this example is very specific: it helps the user understand the energy consumption of a particular behavior (using heating). Connecting the evidence to specific actions helps users understand how to change those actions. It also targets loss aversion: the tendency for people strongly to prefer to avoid losses than to make gains.<sup>26</sup> A message that says “if you don’t do this, a bad thing will happen” is estimated to be nearly twice as powerful as a message that says “if you do this, a good thing will happen.”

---

25 Thomas Goetz, “Harnessing the Power of Feedback Loops,” *Wired*, June 19, 2011, [http://www.wired.com/2011/06/ff\\_feedbackloop/all/](http://www.wired.com/2011/06/ff_feedbackloop/all/).

26 D. Kahneman and A. Tversky, “Choices, Values, and Frames,” *American Psychologist* 39 (4): 341–350, <http://dirkbergemann.commons.yale.edu/files/kahnemann-1984-choices-values-frames.pdf>.

Context is also important in making insights relevant and meaningful (as discussed in “Making Meaning and Enabling Action”). Energy trends might be best explained in terms of the user’s own usage year over year (rises/falls in consumption) or by comparison to neighbors/other similar households. But comparisons can backfire if users don’t perceive the people against whom you are comparing them actually to be similar: for example, comparing the energy usage of someone with three children to their neighbor who lives alone won’t be well received. And if you choose to compare behaviors to other behaviors, choose your comparisons carefully to avoid muddying the picture of what you deem “good” or “bad.” You might think that comparing someone’s carbon consumption from tumble dryer usage to the equivalent of a 600-mile bus journey from San Francisco to Las Vegas would make them think “gosh, that’s a lot.” But if that’s a journey the user would typically expect to take by plane, taking the bus seems like the virtuous option. You can also use comparisons to help frame the consequences of actions. A food tracking service might encourage you to think twice about eating that doughnut by telling you that you’d need to walk six miles to burn it off.

As already discussed, the timing of insights is also critical. If you can catch the user at a time where their motivation to change is high (e.g., when the next energy bill is due) or when there’s an opportunity to engage in the desirable behavior (or to head them off from the undesirable behavior), your message is more likely to be acted upon.

When it comes to giving advice, tone of voice matters. All of us like to make better decisions in our lives, but most of us don’t like to be told what to do. Make the user feel they have a choice, even if you’re steering them toward it. And before you berate them for a “bad” behavior, consider that you might not be aware of a legitimate justification for that behavior. If someone hasn’t walked at all for a week, it might be that they’ve broken their ankle. If someone’s gas usage data tells you that they have an old and inefficient boiler, it doesn’t necessarily mean that they are ignorant of the fact. It might be that they live in a rented property and can’t convince the landlord to replace it. Not every problem is the user’s fault and behavioral change may be only part of the solution. Because of this, it’s worth working to understand the user’s ability to act (e.g., fitness trackers might allow users to specify any health conditions, and energy insight services might ask users to indicate whether they are tenants or homeowners, and which appliances or systems are their own responsibility). Your aim should be to help people set

realistic goals. Designing a “savings target” for an energy insight service might seem like a great idea, but if you find that users expect to save 50% off their bills, that might not be realistic. Likewise, setting the goal of running a marathon with only one month of training isn’t realistic. Allowing users to set these kinds of goals is over-promising and setting them up for failure.

But in some cases, the collective gain is greater than the gain for any one individual. Opower’s average energy saving achieved across their customer base, for both electricity and gas, is approximately 2%.<sup>27</sup> That might not seem like much in the context of an individual’s bill. But across millions of customers, it adds up to a significant carbon saving.

## VISUALIZING DATA

Data visualization is a powerful tool for communicating insights. Like designing for behavioral change, it’s also a huge topic in its own right. It’s beyond the scope of this book to cover in depth. Trina Chiasson and Dyanna Gregory’s free, crowdsourced ebook *Data + Design*<sup>28</sup> is a great place to start. Edward Tufte’s *Beautiful Evidence*<sup>29</sup> and Ben Fry’s *Visualizing Data*<sup>30</sup> are also very useful resources. In this section, we’ll give a brief summary of some key issues.

The data from an IoT system could be displayed on the device itself. Many connected embedded devices, as we saw in Chapter 8, have limited display capabilities, so you’re unlikely to get to display complex graphs on such a device. It might be better to focus on pared-back insights. There is a lack of widely understood conventions for using restricted displays, and you’ll likely need to bake the design for your display into the hardware, so you can’t change the UI later (see Figure 13-22).

If you’re displaying data in a web or mobile app, you’ll have a lot more screen estate to play with. But the oft-repeated message throughout this chapter—focus on the user’s goals for using the system—still holds

---

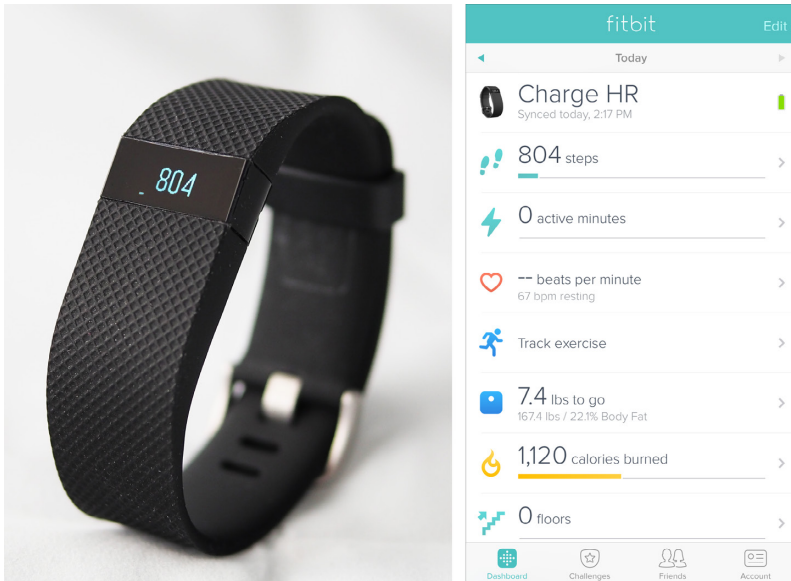
27 <http://opower.com/results>

28 Trina Chiasson, Dyanna Gregory, et al., *Data + Design*, <https://infoactive.co/data-design>.

29 Edward Tufte, *Beautiful Evidence* (Graphics Press, 2006).

30 Ben Fry, *Visualizing Data* (O’Reilly, 2008).

true. The connection between the data and the user's needs must be apparent. Relevant features in the data should be highlighted: just putting numbers and graphs next to each other is not enough.



**FIGURE 13-22**

Fitbit band showing simple step count, with fuller information in the mobile app (images: Fitbit.com)

Statistician and data visualization pioneer Edward Tufte argues that information design—or as he calls it, Analytical Design, should be driven from the question: “How is this design supposed to help people think about the evidence?”

Returning to our heart rate monitor example from earlier in this chapter (Figure 13-15), what are the high, average, and low measurements generated by the heart rate supposed to help the user understand? The vertically aligned line graph and bar chart seem to suggest a relationship between these measurements, but what is that relationship?

The data—or evidence—is present but there is no guidance to help the user think about it. The spread of data unfortunately misses the opportunity to aid the consumer in examining and reflecting on the evidence, and therefore offers little value.



In our heart rate monitor example, if we know the consumer wears the device because he wants to lose weight, we can apply these principles to help him make better sense of this data. For example, we can connect his heart rate and steps, and highlight the intensity of his activities throughout the day. Second, we can compare the length of high-intensity activity in the day with their average. We can even go one step further and compare this user's data with data from other users who share similar physical characteristics and fitness goals. Lastly, we can show causality by connecting trends in activity levels over time with the user's weight. Instead of simply showing the data, we can demonstrate how the system of data connects to the consumer's goal, which will help the consumer think with the evidence.

The nature of your data will also affect the way you visualize it. Are you displaying live data? A visual that includes some animation, like a dial, might be appropriate. Are you showing a continuous stream of very frequent readings, whether live or historical, like electrical power consumption throughout the day? In that case, a line graph would be suitable. Or do you have intermittent, cumulative readings, like a step count or half hourly gas consumption data? In that case, a bar graph is more appropriate. Overall trends should show direction of movement—that is, whether your energy consumption or general activity level is rising or falling. And it's worth highlighting any headline insights and displaying inferences and explanations available from within the graph.

Visualizations aren't necessarily a tool only for end users. The proliferation of sensors and complex networks of connected devices will create a need for service providers themselves to use tools, some visual, to monitor data and activity from around the system to keep it running smoothly.

Visualization guidelines often focus on historical analysis. Data is aggregated into a single set, ingested and processed before visualizations are generated. This takes time. As we saw earlier, IoT data is often at least partly real time and coming from multiple sources. For example, your energy meter display may have a local reading, but also take in aggregated data about other people's consumption, too.

The principles for designing visualizations are still valid whether you're dealing with live data or not. But real-time analysis is harder to do, so you might not be able to deliver the same complexity of insight as you could with historical data. It's one thing to say that, over the last three

months, you spent 20% more on running your air conditioning than similar households. It's quite another to say that your AC is currently using 20% more power to run than your neighbors. That kind of analysis requires much more complex processing. It may be possible, but it will be more expensive to do. The cost would have to be justified in terms of the value to the end user, or your business.

## **USING DATA IN THE DESIGN PROCESS**

Data-driven design has become increasingly popular in web and mobile apps.<sup>31</sup> Using A/B or multivariate testing, software services can push alternative designs out to large samples of users and see which ones fare best with users.

This can be harder to do with hardware. The more functionality and interface design are baked into a physical device at the point of manufacture, the harder it is to make subsequent changes. But where device functionality can be modified in software, it is easier to iterate. The Little Kelham case study, between Chapters 6 and 7, cites the example of a button placed by each home's front door. The designers weren't sure exactly what it should do, but a button by the door seemed like a useful thing to have. The functionality of the button can easily be modified through software once residents have been living in their homes for a while and had a chance to figure out the most useful thing for the button to do.

Data-driven design is often thought of as working with quantitative data: data that tells us what lots of users are doing. Knowing what users are doing can help you identify problems: features that users aren't using or aren't using well. But it doesn't help you figure out what to do about it: for that, you also need to understand why. This comes from qualitative data: observing and talking to smaller numbers of users, as we saw in Chapter 5. Little Kelham is a good example of designers combining knowledge of what users are doing with the system with qualitative insight: observing and learning about how residents use their systems.

---

31 For example, <https://www.ted.com/watch/ted-institute/ted-bcg/rochelle-king-the-complex-relationship-between-data-and-design-in-ux>

When you're designing a data-driven product, it's important to start working with real data as soon as you can. It's hard to optimize the UX around the insights unless you know what they are likely to be.

## Summary

The falling cost of sensing and computation means that there are more and more devices that are capable of capturing, and acting on, data about the world. This provides us with information about the environment, our behaviors, and the devices themselves on a scale that was not previously available. Where data is processed—in a central service or in the edge devices themselves—depends on the nature of the product and can affect the experience of using it. IoT data may be directly measured or inferred, big, small, real-time, historical, and measured at different time intervals. All of these affect what you can do with it, in terms of product function and UX.

The current crop of consumer products on the market can be thought of in terms of whether they capture personal or environmental data, and whether they enable the user to take more informed action or whether the system acts on the data itself.

To create meaningful outcomes from data, designers must consider users' motivations and goals for using the product and provide actionable insights that set data in context. Users' attention is a precious and finite commodity of which we must be respectful when communicating data. Engaging users in the early stages of using a data-driven product can be a challenge, before the product has much data. If you're designing for behavioral change, insights and advice must be sensitively crafted. Visualizations should be designed to help users think about the evidence. Following a data-driven design process can be more challenging when there's hardware involved.



# Iterative Design: Prototyping and Learning

**BY ELIZABETH GOODMAN**

Iterative design—rounds of observing the world, generating ideas, building prototypes, observing the prototypes in use, and returning to idea generation—is the best way to build connected products and services that are not just useful but a joy to use. In previous chapters, we discussed observation and idea generation. In this chapter, we'll discuss prototyping and evaluation, which typically go hand in hand in iterative design processes. Both are necessary for discovering new possibilities and avoiding pitfalls.

This chapter introduces:

- Characteristic dynamics of iterative prototyping and evaluation (page 576)
- Lightweight methods for prototyping initial product ideas (page 582), embodied experiences (page 591), services (page 594), working algorithms and streams of data (page 596), and novel interaction modes (page 599)

It also addresses the following issues:

- What prototypes are for (page 577)
- How to decide what to prototype (page 603)
- Why teams avoid evaluation...and what to do about it (page 607)

## The Necessity of Working Iteratively

In theory, iterative design is very simple: you learn about a situation, you make something to intervene in it, you watch other people trying what you've made, and then you apply what you've learned. Then you repeat as many times as you can. In practice, iterative prototyping is not nearly so straightforward.

Consider the experience of hotspot/router startup BRCK, as told in our case study (between Chapters 12 and 13). BRCK discovered users' difficulties with initial setup after they started shipping their router. The shipping schedule did not give the team enough time to manufacture new hardware or plastic cases. To move quickly, the team instead revised the packaging, initial instructions, and online support.

The Sifteo Cubes, a connected game platform, faced a similar problem. The award-winning cubes had an elegantly simple battery insertion mechanism that first-time users found “not so obvious,” said founder David Merrill. Not being able to insert the battery meant not being able to use the cubes at all.

Like BRCK, Sifteo discovered the problem too late to fix the root cause. Altering the battery compartment itself would require time and money Sifteo didn't have to spare. So Sifteo, like BRCK, took a faster and cheaper path: redesigning the printed instructions, packaging, and instructional videos (Figure 14-1). Quick rounds of experimentation led the designers to a transparent sticker with a blue arrow pointing to the battery insertion location. Then they set about testing whether the arrow solved the problem. The team tested 10 different arrow versions before finding a size, color, and placement that reliably prevented first-time difficulties (Figure 14-2). (Chapter 12 has more on key interactions such as initial setup.)

The experiences of BRCK and Sifteo remind us that no team, no matter how brilliant, gets every product decision right the first time. The only question is whether teams factor prototyping and evaluation into the timeline—or whether the need for prototyping and evaluation come as a surprise.



**FIGURE 14-1**

A video walks first-time users through battery insertion (video: David Merrill/Sifteo); view the video at <https://www.sifteo.com/manual>



**FIGURE 14-2**

Blue arrow stickers indicate how to line up battery compartment and lid (image: David Merrill/Sifteo)

## Using Prototypes to Answer Questions

*The value of prototypes resides less in the models themselves than in the interactions they invite.*

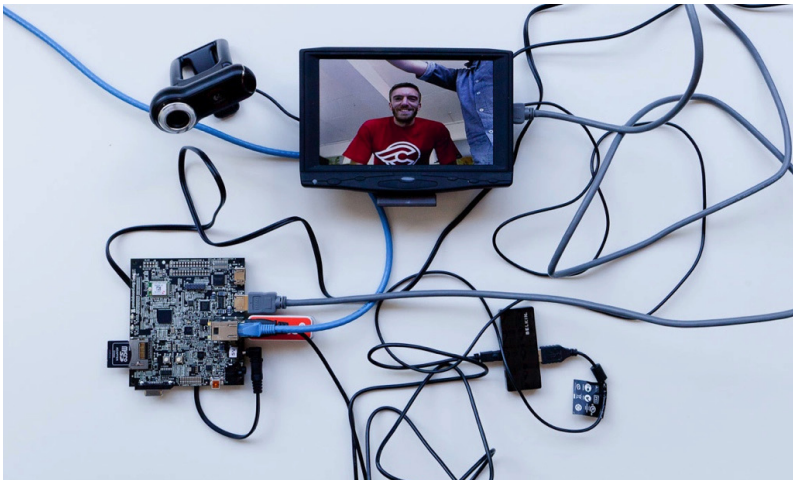
**MICHAEL SCHRAGE, SERIOUS PLAY<sup>1</sup>**

---

<sup>1</sup> M. Schrage, *Serious Play: How the World's Best Companies Simulate to Innovate* (Boston, MA: Harvard Business Review Press, 1999).

Prototypes are experimental models—objects crafted to help answer specific questions about the project. As models, prototypes are deliberately simplified pictures of the project. Their constraints help us concentrate on what matters most, rather than distracting us with less important details. As experiments, prototypes are most helpful when we let them surprise us.

The goal is to make the right prototypes for *your* project. When design firm BERG worked with Google to instantiate voice/video chat services in a standalone device, the firm ran two parallel prototyping efforts, for what they called Connbox.<sup>2</sup> The first effort evaluated technical feasibility. *Would it be practical to manufacture such a device?* The team started by making a functional, but fragile and messy, “end to end” software and hardware package (Figure 14-3). To ask, *What would life be like with a mostly-on video connection?*, the team made the prototypes more robust, with more durable cases (Figure 14-4), and started to try them out (Figure 14-5).



**FIGURE 14-3**

An early prototype proves the team could make a functional object, but is hard to deploy (image: BERG)

---

<sup>2</sup> Described in: M. Jones, “Connbox: Prototyping a Physical Product for Video Presence with Google Creative Lab,” 2011, <http://berglondon.com/blog/2013/02/26/connbox/>.





**FIGURE 14-4**

A simple case helps make the prototype more robust, while minimizing attention to the industrial design (image: BERG)



**FIGURE 14-5**

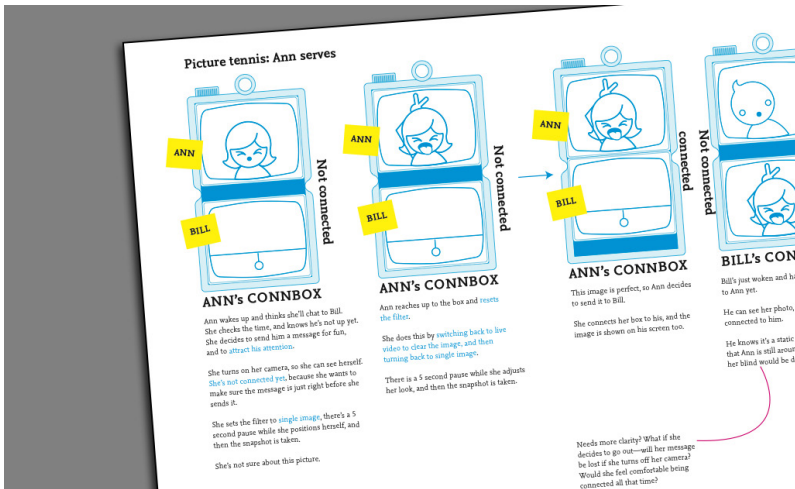
The designers install the encased hardware in their studio (image: BERG)

Interaction and industrial design prototyping, informed by life with the functional prototype, next explored how to make users feel comfortable with a mostly-on video appliance. Drawing on human-computer interaction research into “beautiful seams,”<sup>3</sup> the designers asked themselves how “to make the state of the connection as *evident*

---

3 M. Chalmers, I. MacColl, and M. Bell, “Seamful Design: Showing the Seams in Wearable Computing.” *IEE Eurowearable*, 2003 (pp. 11–16). doi:10.1049/ic:20030140.

as possible, and the controls over how you appear to others simple and direct.” Storyboarding a scenario (Figure 14-6) forced them to work through the complexities of keeping and discarding: *Should the device be attention-getting enough to interrupt an ongoing activity?* Through iterative cycles of storyboarding, the team established more key features for control and directness, such as gestural interactions with a “do not disturb blind” to partially block what the other box sees.



**FIGURE 14-6**

Interaction design storyboarding spurs more questions (indicated by red line) (image: BERG)

Next came “looks like” industrial design prototypes (Figure 14-7), which asked *How might mechanical controls such as dials and slides make the connection state more visible and control more direct?* But, as Matt Jones writes:

This is not a simple translation between a software and hardware behaviour, it's more than just turning software controls into physical switches or levers.

It involves choosing what to *discard*, what to *keep* and what to *emphasise*.

The final answer: a looks-like prototype that was still minimal, but that used color (e.g., orange for video, blue for connection, green for audio) to emphasize the current state of the software and how the user can mechanically change it.



**FIGURE 14-7**

Simultaneously, industrial design “looks like” prototypes consider how to make system status more visible and control more direct (image: BERG)

The Connbox process shows us that decisions about what and how to prototype are situational. As teams work from initial ideas to more polished proposals, they decide as they go what types of prototypes will help them. Different lines of prototypes may progress in parallel (as with Figures 14-4, 14-6, and 14-7), then merge into a single polished whole. Some prototypes, like the scenarios in Figure 14-6, are conceptually influential but are not meant as specifications for final designs.

The following sections will list some typical questions and introduce ways to answer them. However, there is no automatic “best choice” at any point. In deciding how and what to prototype, we like this advice from design scholar Youn-Kyung Lim and her colleagues:

The best prototype is one that, in the simplest and the most efficient way, makes the possibilities and limitations of a design idea visible and measurable.<sup>4</sup>

4 Y.-K. Lim, E. Stolterman, and J. Tenenberg, “The Anatomy of Prototypes: Prototypes as Filters, Prototypes as Manifestations of Design Ideas,” *ACM Trans. Comput.-Hum. Interact.* 15(2): 1–27, doi:10.1145/1375761.1375762.

The methods in the next sections are ordered roughly in terms of technical complexity. We are not, however, implying any necessary chronological sequence. Instead, we see these sections as a sort of recipe book from which you can select the techniques most appropriate to your questions and adapt them to your own taste and resources.

The chapter will *not* address the mechanics of exploration and evaluation. Any usefully detailed explanation of evaluation methods would likely double the length of this already long book. Instead, this chapter will delve into the prototyping concepts and tools that make evaluation exercises easier to integrate into your project.

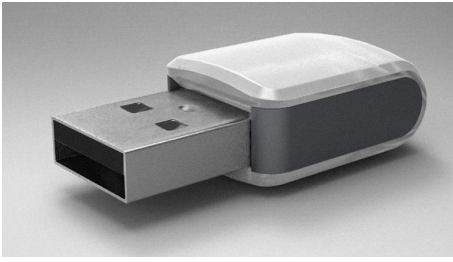
### **WHAT DOES THE WORLD LOOK LIKE WITH YOUR PRODUCT IN IT?**

If you're designing a product or service that radically differs from what already exists, how do you get informed comments from decision makers? This is particularly difficult with connected products, which often involve new relationships among existing systems and infrastructures, and new forms of data capture and analysis. You want people to base their responses on a clear and concrete vision of the device in context. However, if what you're proposing is vastly different from their past experiences, evaluators will likely need help imagining the world with such a new product in it. At the same time, you don't want to spend too much time detailing interfaces and interactions for a product concept that you might end up discarding.

The solution is a high-level, but convincingly real, representation of a future in which the product is *already in use*. Here, we'll discuss three quick ways to evoke future products in use: collages, media from the future, and video prototypes.

### **COLLAGES**

Many disciplines, from industrial design to architecture, create realistic mockups. Generally, these artifacts are created late in the design process to help clients or other decision makers settle on a final design before moving to manufacturing/building. Often, as with industrial design (Figure 14-8), these "concept designs" are shown in isolation from the environments of use and systems within which they work.



**FIGURE 14-8**

Industrial design concept rendering for the Blink(1), a programmable notification light; the Blink(1) requires a USB connection to a computer that is not shown in the rendering (image: ThingM)

In order to get a handle on systemic complexity, designers of connected products often reverse these established practices. They make semi-realistic mockups early on in the process in order to help nonexperts imagine unfamiliar devices and usages. And they make documents that situate physical objects *in context*—within the environment of use, as a part of other services and systems, and perhaps as a channel for existing media.

So teams often need to make trade-offs in effective early concept communication for connected products. On the one hand, they need to convey the full complexities of a product–service system, often to project outsiders who don’t share the team’s background or assumptions. On the other hand, the goal of early concept communication is to use concepts as springboards for discussion: combining elements, deleting elements, rejecting some, keeping others, generating new ideas, and generally exploring what kinds of products and uses make sense. To do that, you need to generate and represent many ideas in a short period of time. So in early deliverables, designers are often trying to convey, simultaneously:

- A realistic-seeming representation of the product in use
- High-level mapping of relationships to systems, platforms, and infrastructures
- That despite the semi-realism, the concept should be taken as open and unfinished

...without spending too much time on any one document.

“Story collages” is an umbrella term we’ve invented for two lightweight concept communication techniques—storyboards and product/service visualizations—that product, service, and environment designers successfully use to tell stories about interconnected products and systems. Mixing representational styles is the key to communicating newness, “everydayness,” and openness in a single artifact.

## PRODUCT/SERVICE VISUALIZATIONS

One of the first steps in a new project is often translating inspiring rhetoric into concrete specifics. The blue-sky technological visions are exciting, but it’s often hard to see exactly how one might put them to work. For example, in working with a global consortium of city managers, the design and planning firm Arup needed to put “smart cities rhetoric into something plausibly everyday,” writes designer Dan Hill, for a generative design workshop on future urban forms and services. Their goal was not to convince the attendees to implement any given idea, but to use the concepts as “pivots for conversations” about the specific aspirations and concerns of each city.

To fuel the conversation, they created what we’re calling product/service visualizations—single-page mockups of key forms, relationships, and interactions. These mockups montage photographs of existing sites with renderings of the concepts. The interfaces are a little rough and generic and the renderings are deliberately quick and imperfect. The images aren’t even sufficient—viewers have to read the annotations to grasp what’s going on. But perfection isn’t the point. The point is to condense a complicated set of interactions and systems into a *specific* and *concrete* image that conveys both problems and possibilities.

That’s why we call them product/*service* visualizations: they make visible the relationship among products, services, environments, and systems. For example, Figure 14-9 shows relationships among a smart bike garage, screens, dynamic data, riders, and passers-by during the day and night.

If they are obviously up for debate, such images don’t have to present desirable or even feasible concepts. The Arup team, for example, deliberately included some flawed concepts so that the workshop participants could debate them. They turned the concepts into posters (Figure 14-10) to make them easy to pass around a table.



**FIGURE 14-9**

Product/service visualization for a shared urban bicycle service (image: Dan Hill)



**FIGURE 14-10**

A product/service visualization prompting discussion at workshop as a print-out (image: Dan Hill)

## Storyboards

We won't go into much detail about how to make storyboards from written scenarios. First codified by John Carroll and then popularized by designer Alan Cooper in his team-written *About Face* design textbooks, software developers have used scenarios and storyboards widely for decades.

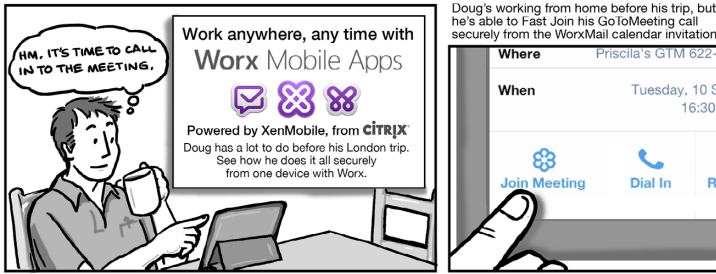
One reason storyboards are useful for early concept generation is that they incorporate multiple forms of graphic communication gracefully. In telling stories about future products, we are often describing not just the immediate context of interaction, but how systems work together—how a car, for example, could play streaming radio, sense engine failure, turn the radio off, warn the driver, and push directions to the nearest garage to the driver's mobile phone. Like comic books, storyboards draw together not only images of people and objects, but diagrams, maps, and symbols. Making a storyboard forces teams to work through how those systems might fit together.

Storyboards are also useful, strangely enough, because they are obviously *not real*. As with software development, storyboards allow us to sketch roughly what we do not want detailed and to skip parts of the story we have not yet worked out. The storyboard in Figure 14-11 mixes corporate icons and screenshots of working software with cartoon figures and interfaces. The mix of cartoons, screenshots, and hand-drawn interfaces successfully communicates which parts of the product/platform system are already working, and which ones are up for debate. Another typical technique, borrowed from architecture, is to superimpose flat silhouettes of people and devices on real background images.

## Media from the future

One way to evoke a plausible, everyday future is to imagine how outsiders—from journalists to advertisers—might portray it. Another way Hill brought specificity to the broad idea of a “smart city” was to fake an article in an invented local newspaper (Figure 14-12) detailing a future filled with new urban infrastructures and services.





Doug's working from home before his trip, but he's able to Fast Join his GoToMeeting call securely from the WorxMail calendar invitation.

Uh-oh! The presentation he's giving in London isn't ready!



To keep track of his team's progress, and to get files easily while he's traveling, Doug downloads Podio and Sharefile apps from the Worx Store.



Photos of the Caribbean © Disney, Apple iTunes logo © Apple Inc.

**FIGURE 14-11**

Citrix Worx storyboard (image: Deb Aoki)

As Hill says, writing such an article is “a useful test to consider whether your design, service or strategy, could be plausibly conveyed in a newspaper article. It’s also a good test for the designer to be able to convey their ideas through this kind of writing.”<sup>5</sup>

5 D. Hill, “Sketchbook: Melbourne Smart City, for City of Melbourne/C4o Cities” (incl. a note on why it’s easier to crowdsource a revolution than a light-rail system), August 26, 2011, <http://www.cityofsound.com/blog/2011/08/melbourne-smart-city-c4o.html>.

**Special edition**

# How we became a smart city

Melbourne has transformed the way it lives, works, and plays

By Mark Sharp and Abi Davis

MECHELLE is waiting for the greenlight 0:23 from St. Kilda to the City but you wouldn't know it.

She glances sidly at the iPad on the table in front of her and taps the last of her espresso. The iPad's screen indicates a small green dot moving slowly down a map of the area.

The app, powered by the City of Melbourne and State Government's joint open data initiative, indicates all her mobility choices, organised around her daily patterns. In this case, it shows her she's got two minutes to get to the tram stop a few metres away, and that the tram is only half full.

"It's funny," Michelle says. "This is exactly how some tram services I had four because I feel in control of it. Because I have the information at hand, I can make the right decisions. For me."

This is a sentiment increasingly echoed by many Melburnians, and not just limited to transport. Across many walks of life in the city, and many daily activities, the city of Melbourne suddenly feels like its infrastructure is as contemporary as that of Singapore, Seoul and Shanghai to the north.

As a result of the initiatives in place over years ago by the then Lord Mayor Robert Doyle, Melbourne

can genuinely claim to be a smart city.

Very early intervention was to connect together the city's various public transport systems via the Myki system, which can now be accessed via smartphones as well as smart cards. While investment in public transport infrastructure has also increased, the shift in the experience to the public what has had Melburnians returning to public transport in droves. Trams, buses and trains are now comfortable, well-lit-on-experience which other services for commuters and others that private cars simply cannot match.

Yet congestion on the roads has also abated due to this up-tick in public transport use and smarter systems managing traffic flow and parking in the city.

The effect on the city's carbon footprint is immediately visible in the landmark installations installed and around many city streets.

These displays—at once public art and urban interface—have become hugely popular with visitors, tourists and residents, and variously provide information and enticing insights into how Melbourne is performing. A particular favourite seem to be the sound installations along the Yarra, which not only indicate noise levels in the city, but also mash-up real-time, the number of different languages being spoken at key points in the city, which prompts the

sounds of other connected world cities from time to time, chosen on the basis of their real-time internet connectivity with the city. Walk past and you hear snippets of Shanghai and Seoul, The Beatles and Lady Gaga, Flinders Lane and the foxy from A&M Park. It liberally speaks of Melbourne in the 21st century.

The roll-out of free public Wi across the city combined with some bespoke Council-run working hours, has led to people being able to do much of their work closer to where they live, with numerous benefits in terms of health, usability, traffic and carbon.

Importantly, understanding carbon had been brought to the forefront of pooled, fees throughout a series of activities and services in Melbourne. Some of these used on smart meter data, while other numbers from the age numbers of their sensors, which are easily identifiable in City of Melbourne levery and dotted around the city.

It's unclear where all this might end. Municipal waste collection has been publicly improved due to better data. Complaints to the City are down, as many citizens are able to do themselves, where clear how quickly and cheaply through their own neighbourhood—relative to yet the complaints that are made can be tracked in detail and are used strategically by the city.

Activity on city streets has increased, thanks to the support of the city's

already-excellent street furniture and urban design, as well as programs like the 100 Oases project, which had seen the city's sport infrastructure heavily used and carefully maintained by a connected "Green Army" of volunteers.

Numerous other social innovation campaigns are being coordinated online, many of which are powered through the City's web services and open data feeds.

The Real-time Melbourne city model has become the core interface for citizens, app-makers, government strategists, and other cities, all keen to learn from how Melbourne has wrangled its data into smooth-running public services and urban experiences.

This combination of better city services and better visualisations conveying the way the city is performing has helped make real progress against the city's previously unenviable carbon profile.

While some have questioned whether the Melbourne Smart City program should have been a municipal priority, few have actually criticised the approach in practice, particularly when it became clear how quickly and cheaply through their own neighbourhood—relative to yet the complaints that are made can be tracked in detail and are used strategically by the city.

Through a considered combination of these technologies, and people-centred and

joined-up governance perhaps unusual to Australia, Melbourne managed to demonstrate how a series of small-scale, exemplar projects might quickly coalesce into a better public network, and become something bigger than the sum of its parts.

While it clearly enables a more efficient form of government, it also seems to suggest an expanded role for government into places the private sector would not reach. Melbourne's City of Melbourne estimates that the City will have the public data—data that is not to be confused with the alternative, Smart and professional handling of privacy and the ethics surrounding data capture has probably been instrumental here.

This public data has then been made available to app developers and citizens—all real-time Melbourne—such that the average person in Melbourne has access to an extraordinary rich set of apps that provide new interfaces and activities onto their city.

It has also meant that Melbourne is now recognised globally as a key city in terms of smart development and other innovation. The sector here is thriving and it says that an obviously beneficial to citizens over and above its direct economic impact.

Michelle waves as she steps up onto the tram, pleased that her app delivered accurate results. Not least because she designed it herself.

▶ Continued page 6



## FIGURE 14-12

Fake newspaper article for the Smart Cities project (image: Dan Hill)

### Video prototyping

Video prototypes demonstrate a concept in action. What differentiates a video prototype from the polished “concept videos” we often see online is speed of execution. Unlike final concept videos, video prototypes are typically quick to make, low budget, and rough. All you need to make a video prototype is a digital camera (even a mobile phone will do) and editing software such as iMovie or even Camtasia. The point is to help yourself, and other people, grasp the central elements of the idea rather than wowing them with Hollywood-polished special effects.

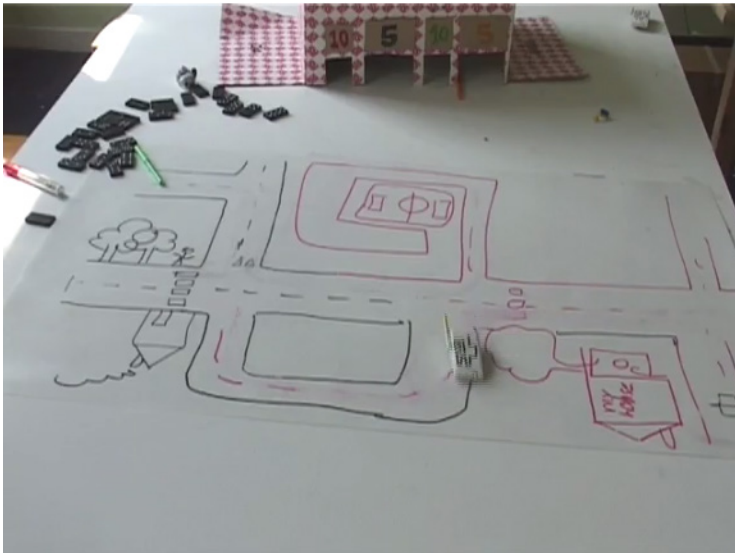
Video prototyping is a central tool for designing connected products because it more faithfully represents multisensory interaction. Activities involving gestural interfaces and robotic movement are hard to represent in still images. It requires imagination—and previous

experience with those technologies—to translate a sketched arrow on a page to the hand gesture it symbolizes. With video, you can simply show a hand moving.

Video prototyping builds on the prototyping techniques we introduced previously: making product–service visualizations help you select key moments to dramatize; storyboards help you put together the story to tell; future media help you work out the aesthetics of your imagined future. Is it sketchy and playful, like Figure 14-13? Is it homey but serious, like Figure 14-14?

Combined with voice narration and titles to explain the action, common ways to fake working technology include:

- Live-action role-playing with props (Figure 14-13)
- Stop-motion animation with objects in the place of people (LEGO figures work well)
- Still montages of paper prototypes and photographic images (Figure 14-14)



**FIGURE 14-13**

Sketch-a-Move, a toy concept video. Small magnets under the table create the illusion that players are controlling the movement of toy cars by drawing on them; view the video at <http://vimeo.com/5125096> (image: Anab Jain, Superflux).



**FIGURE 14-14**

Video prototype for the Economizer, a home electricity use monitoring project from Cooper Design for the Environmental Defense Fund; view the video at <http://www.cooper.com/journal/2008/12/economizer.html> (image: Cooper)

Our one warning: with the advent of high-quality consumer video cameras and software, you can make video prototypes that are *too convincing*. If your product is entirely speculative, real-looking video can mislead your audience into believing you can deliver working technology immediately. That's why it can be useful to include obviously “sketchy” elements like the hand-drawn portions of the Economizer video. Moreover, video prototypes usually tell best-case, *idealized stories*. These represent wishful thinking, not prediction.

### **WHAT DOES USING A PRODUCT FEEL LIKE?**

By “feel like” we mean spontaneous, emotional experience. What does it *feel like* to be told, while driving, that your car is about to have an engine failure? What does it *feel like* to realize that your thermostat has decided to cool down the house without asking first? When introducing radically new product types or modes of interaction, it's particularly important to ask these sorts of questions early on. You certainly can't draw on previous experience with similar products to answer them.

However, such questions are difficult to answer with functional prototypes. If you're prototyping a smart watch that you expect someone will wear *everywhere*, you need to ask: what does it feel like while working out? In a meeting? During a romantic evening out on the town? If you wait to evaluate product experience until you have a working prototype robust enough to survive the gym, you have probably sunk too much time into one direction.

## Experience prototypes

Instead of simulating form or function, *experience prototypes* simulate key sensory factors of use—perhaps an environment (e.g., an elevated hospital bed), a psychosocial situation (e.g., getting an engine failure alert while driving), or a physical condition (e.g., blindness).<sup>6</sup> Marion Buchenau and Jane Fulton Suri of IDEO coined the term. In their words, experience prototyping allows “designers, clients or users to ‘experience it themselves’ rather than witnessing a demonstration or someone else’s experience.”<sup>7</sup> There are many different forms of experience prototyping. Here we’re interested in learning from theater: how to evoke the sensations of use with low-tech or inexpensive means. A full-size appliance can be mocked up from cardboard boxes (Figure 14-15); a pager can stand in for an Internet-connected heart defibrillator.

Consider that networked defibrillator, still a classic example of the benefits of experience prototyping. Some questions the designers needed to answer were: “What is it like to be a defibrillating pacemaker patient? What is it like not knowing when and where defibrillating shock will occur? How does that affect people’s everyday life?” To answer them, the designers decided to simulate the experience of wearing the defibrillator by carrying pagers for a weekend. Each time they got a page, they imagined that they had received a defibrillating shock strong enough to cause them to fall. Then they recorded the circumstances and how they felt about receiving a “shock” in that moment. Of course, in order to set up the activity the designers had to know already know a little bit about the feelings and experience of heart defibrillation patients. That’s where initial exploratory research came in.

---

6 We mention similar techniques in Chapter 6 as “immersion tools.”

7 M. Buchenau and J. F. Suri, “Experience Prototyping,” in *Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (New York: ACM, 2000), 424–433, doi:10.1145/347642.347802.



**FIGURE 14-15**

Experience prototyping with cardboard props standing in for appliances and people (image: D-LABS)

It's telling that the technique emerged in part as a response to the challenges of designing an Internet-connected product. Experience prototyping is an elegant way to investigate the characteristic dynamics of connected products and services—encounters that the user does not fully control (like a defibrillator) or that may depend upon multiple components in the built environment (like an appliance).

### **Wizard-of-Oz demonstrations**

In Frank Baum's book *The Wizard of Oz*, the heroine faces a terrifying, ghostly apparition—which is actually a projection controlled by a hidden human.

Similarly, in Wizard-of-Oz prototyping, a hidden human simulates the actions of a machine. It's a type of experience prototyping.

In Figure 14-16, for example, a demo from Ericsson Labs of a shared interactive space based on touchscreen interactions relies on two actors in another room (Figure 14-17). One, visible to the audience, speaks to

the demonstrator. The other, off-screen, watches the action and triggers on-screen actions based on the gestures he sees. The two screens aren't even on the Internet—they're just connected by a long cable. The screen interface itself is just a rough mockup designed to suggest functionality without much polish.



**FIGURE 14-16**

Wizard-of-Oz prototype of a video-driven collaboration service (image: Ericsson User Experience Lab)



**FIGURE 14-17**

Behind the scenes of the Wizard-of-Oz prototype, the person at the laptop manually triggers animations as the demonstrator speaks (image: Ericsson User Experience Lab)

The advantage to the method is that it provides a convincing demonstration of unproven technologies relatively quickly. Even those who know that the experience is faked will often suspend their disbelief long enough to give genuinely felt feedback. The disadvantage is that Wizard-of-Oz prototypes that rely on triggering preprogrammed actions, as with the previous example, are inflexible. They require choreographed demonstrations rather than open interaction.

Marcus Nyberg of Ericsson Research stages such “immersive presentations” in order to stimulate emotional, not just intellectual, reactions to proposals early on. He writes:

For services, including several interconnected applications, devices, objects, and so on the method is particularly relevant as implementing these concepts as functioning prototypes would take a large amount of resources as well as time. And perhaps requiring all this before you know whether you are on the right track or not.<sup>8</sup>

### **HOW DOES A PRODUCT SUPPORT A SERVICE (AND VICE VERSA)?**

The phrase “connected products *and* services” is a bit misleading. From washing machines to running shoes, many products can now be monitored over the Internet. By the same token, newly connected objects are transforming traditional services. The FedEx SenseAware device, which we described in Chapter 4 (Figure 4-17), monitors package location and environmental conditions such as temperature. It adds supply chain management to the original service of package delivery. Connected objects, then, become meaningful *as components of services*.

Service design is an emerging field, but most practitioners agree that it concerns how interactions with touchpoints (interfaces and components) and channels (streams of media) change over time. (See Chapters 6, 9, and 15 for more on service design.) Beyond product hardware and software, service design topics for prototyping and evaluation could include:

- Unboxing and/or setup experience (including instructional materials)

---

<sup>8</sup> M. Nyberg, “Smoke and Mirrors (and a couple of ideas),” December 13, 2012, <http://www.ericsson.com/uxblog/2012/12/smokes-and-mirrors/>.



- Customer support mechanisms during use
- Support for environmentally responsible disposal and recycling

There are some specific aspects of service design that we consider elsewhere in the book, such as interoperability (Chapter 10). Here, we're more interested in providing some prototyping tools to help you work out how any one product or interface fits into more complex services before moving to working prototypes.

Our main recommendation is to work both from the bottom up and from the top down—that is, to move between considering the service as a whole and addressing individual interfaces or components. The key is working quickly and at a high level to avoid getting stuck in the details of any individual component. One good tool is Business Origami, a method invented at Hitachi Design Center.

Business Origami is a game-like activity that uses paper cut-outs to rapidly and iteratively work out the relationships among components at various phases of engagement with a service. Those components should include:

- *Touchpoints*, or the parts of the service with which users interact—such as a bike rental kiosk, a mobile app, rental payment management systems, and even paper receipts.
- *Places* or *locations*, such as a bank or a car.
- Important *actors* for each phase, such as buyers or technicians.
- Key *stakeholders*, especially if you are entering a heavily regulated domain such as healthcare or transportation.

Paper cut-outs stand in for potential service components. As participants list likely components, they place the shapes on a paper-covered tabletop (Figure 14-18). As participants talk through what the service does, they collaboratively sketch and label relationships among the components. The bike, for example, would get linked to the kiosk. The mobile app might be linked to the payment system.

Walking through various scenarios prompts participants to add, delete, and move around the paper cut-outs, redrawing the sketched-in relationships as they do so. Representing service components with paper cut-outs is especially helpful because paper is cheap and easy to work with. Multiple people with different levels of technical skill and interest

in the project can interact with different cut-outs at once. When the activity is done, the cut-outs remain as a tangible map of the service and its components.



**FIGURE 14-18**

A team uses Business Origami to work through a service for crowdfunding volunteer activities (image: Nearsoft, Inc.)

Later in the process, evaluating prototypes of individual components might mean revising other parts of the service—and returning to the origami. The smart bikes might end up needing a publicly visible availability dashboard, for example.

### **HOW DO WE PROTOTYPE WITH DATA?**

As we wrote in Chapter 13, data is as much a part of the product as the hardware and the service behind a connected product. Subtracting data from any connected product renders it useless.

To explore what using a connected product feels like, you'll almost certainly need to prototype some aspects of producing, sharing, modeling, and acting upon data. As usual, our advice is to limit what you build. If you let it, prototyping data-infused products and services can engage nearly every form of computer science and mechanical engineering, from sensing and networking to algorithm design and machine learning. Your job as a UX designer, however, is to investigate how people

*experience* data-driven interactions in order to inform business strategy and engineering decision making. You need to figure out how people might *create meaning from data*.

Again, as usual, our recommendation is to spend your time and money making many lightweight prototypes rather than one fully featured one. That being said, you still need to understand the design, engineering, and business questions at stake so that you can decide how your prototype can help answer them.

Chapter 13 discusses designing with data in much more detail. However, UX prototyping questions generally fall into four categories:

### *Sensing*

Translating phenomena outside the device into system-usable data.

### *Sharing*

Transmitting and receiving data from other system components. Is your system closed, circulating data only within itself? Do you need external data sources, such as user profiles, to make sense of internal data? If your business strategy involves selling user-generated data, how are users likely to respond? How might you ask for permission before collecting behavioral data, and how might you inform the subjects of data collection about how you use it?

### *Modeling*

Assigning the data to appropriate categories. For example, an activity tracker isn't literally counting footsteps. Instead, it's running data from accelerometers through an algorithmic model and comparing clusters of data to activity profiles. "Footsteps" happen to be salient to activity tracking. Another project might take the same data and try to identify dance moves. What categories might be meaningful to various participants in the product ecosystem? What kinds of categories or models are likely to be sensitive or controversial? We cannot emphasize enough the necessity of evaluating, early and often, the kinds of inferences you are making about people, how you are using those inferences, and whose interests you are pursuing.

## Responding

Using categorized data to act appropriately. How will your system respond to the data it collects? What data does it need to act appropriately? What does “appropriately” mean to various project constituencies?

### Canned data

Archived, or “canned,” datasets let us speed up, slow down, and replay complex interactions that might change over time. Typically, there are four reasons to use canned data:

- To speed up slow-moving interactions into more easily testable “time lapse” visualizations
- To slow down zippy processes so humans can work with them in detail
- To combine interactions (e.g., initial sign-up and updating the system) that in reality are distant in space and time into artificial but more easily testable unified scenarios
- To evaluate the *feel* of data-driven features before putting time and effort into integrating a live data source

Data platforms (at the time of writing, one of the best known is Xively, but you might find that others better suit your needs) can help with the first two goals by providing canned streams of data to play back at various rates, like a timelapse or slow-motion film. The third goal, however, often requires some fakery.

### Faking it

Data-driven UX depends on algorithms in the background that users might never notice, much less control. So UX needs to drive algorithm development and implementation, not the other way around. Early prototyping should focus on understanding what people need from the system and what the system needs from the people who work with it... not on solving hard computer science problems.

Wizard-of-Oz demonstrations are an excellent way to fake what you don't want to make. Speech and image recognition, navigating a robot around obstacles, modeling human sentiment...many interesting tasks

are easy for humans, but hard for computers. The finished product might have those features, but making a working prototype while testing out multiple concept directions will waste engineering resources.

You can also fake working algorithms quickly by mocking up the results. Trying to imagine a wearable pollution monitor? Take a mobile phone, load up a series of fake visualizations of escalating pollutant levels, and attach the phone to a backpack. Now walk around and watch the pollution levels rise. How does seeing those visualizations make you feel? Do other people find the visualizations on the backpack useful or comprehensible? Is attaching a pollution monitor to the wearer's *back* really a good idea? There's no need to struggle with getting access to the right data and implementing a visualization algorithm if the initial experiment isn't interesting.

However, we warn you: as with video prototyping, one's imagination can easily run ahead of what's technically possible. Unless you are working on a very speculative, blue-sky proposal, consult with experts *first* about what's realistically possible in the project timeframe before getting too attached to your fakery.

## **HOW WILL PEOPLE INTERACT WITH AN INTERFACE?**

Interactions with connected products and services often move beyond the traditional form factors and mouse/keyboard inputs. From wall-size environments to miniature wearables, screens take on new sizes and shapes. New products on the market show a dizzying array of tangible, tactile, haptic, and sound-based input and output mechanisms. Even the very concept of "interaction" is mutating. The early days of interaction design assumed a human interacting with a machine. Real-time data streams and ever more sophisticated algorithms now drive interactions among machines, with computers learning how to make semi-autonomous decisions. If good design lies in crafting the details as well as framing the initial concept, then diagrams or storyboards won't do. We need ways to prototype multisensory interactions quickly and cheaply.

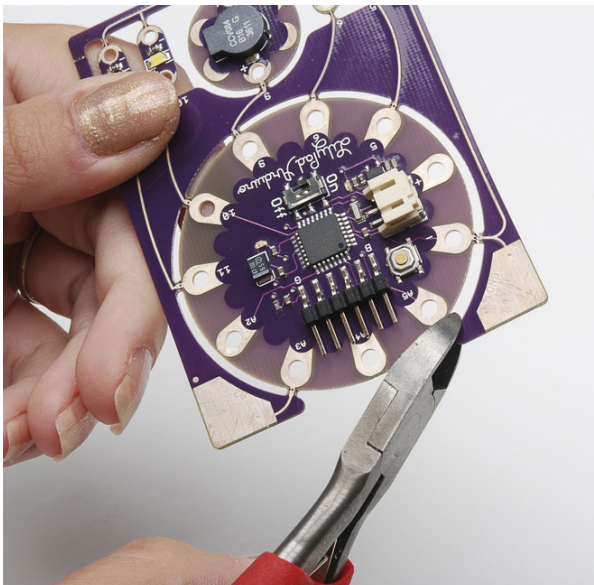
### **Inputs and outputs: beyond the screen, keyboard, and mouse**

Over the past 10 years, prototyping nonscreen-based interactions has gotten substantially easier. The introduction of the Arduino in 2005 led the way. Now there are many new toolkits, such as the Raspberry Pi (launched in 2012), that reduce the level of technical expertise required

to prototype new forms of devices and nonscreen-based interactions. But the rule of thumb still applies: if you need a “works-like” prototype, be picky about what you prototype and build as little from scratch as possible.

**Adapt a working device.** The easiest way to prototype interacting with a device is to find a working consumer electronics device that can imitate what you’re imagining. Tape a tablet to a refrigerator to simulate gestural interactions with a smart fridge. Record a friend talking and have a partner play it back on cue to simulate speech-based interactions (see “Wizard-of-Oz demonstrations”).

**Connect toolkit components.** If you can’t find an existing device that can even vaguely approach what you imagine experiencing, it’s time to start working with toolkits. The key is finding pieces—e.g., toolkits and components—that fit together with only reasonable adjustments. The rise of what’s often called “maker culture” has supported a bunch of websites, such as Sparkfun ([www.sparkfun.com](http://www.sparkfun.com)) and Seedstudio ([www.seedstudio.com](http://www.seedstudio.com)) that sell basic kits such as the LilyPad ProtoSnap wearables package (Figure 14-19) and provide the tutorials (Figure 14-20) and community forums to help beginners get started.



**FIGURE 14-19**  
ProtoSnap—LilyPad Development Board (image: LilyPad)

For readers who, like us, don't have an electrical engineering degree, the key to getting started is finding a supportive community—sometimes online, sometimes in person. Depending on where you live, you may have a local hacker/maker space that can teach you the basics. Or look for distributors that provide readable documentation well as detailed tutorials for beginning prototypers.

The screenshot shows the Sparkfun website interface. At the top, there is a navigation bar with the Sparkfun logo and links for SHOP, LEARN, A/V, FORUM, and DATA. Below this is a secondary navigation bar with links for START A PROJECT, EDU BLOG, RESOURCES, TUTORIALS, CLASSES, CALENDAR, WORKSHOPS, and CONTACT. A search bar is located on the right side of the navigation bar. The main content area features a breadcrumb trail: HOME / TUTORIALS / PROTONAP LILYPAD DEVELOPMENT SIMPLE HOOKUP GUIDE. The title of the page is "ProtoSnap LilyPad Development Simple Hookup Guide". Below the title, there is a section for CONTRIBUTORS, listing JIMBO, and a FAVORITE counter showing 0. There are also social media sharing icons for Facebook, Twitter, and a general share button. The main content area is titled "Introduction" and contains a short paragraph of text. To the right of the text is a table of contents with the following sections: Sections, Introduction, Hardware Explanation, The ProtoSnap Circuit, Powering It Up, Uploading an Arduino Sketch, Raygun Preparation, Unsnapping and Tacking, Sewing the Positives, and Sewing the Negatives. Below the text is a photograph of the ProtoSnap LilyPad Development kit, which is a small circuit board with various components and a white LED.

**FIGURE 14-20**

A LilyPad ProtoSnap tutorial at Sparkfun.com demonstrates a simple first project with the kit in detail, <https://learn.sparkfun.com/tutorials/protosnap-lilypad-development-simple-hookup-guide> (image: Jimb0 and LilyPad)

## New screen interaction paradigms

Some screen-based interactions require less prototyping effort than others. If you are designing apps or websites for existing consumer electronics devices, much of your work has already been done for you. You can draw on premade platform features, design frameworks, and component libraries. With the basic infrastructure already in place, writing code is cheap and easy. Skilled frontend coders can often move directly from paper sketches to low-cost interactive prototypes. And indeed, that's how many interaction designers and developers like to collaborate.

However, designing screen interfaces for *new* devices is a different beast altogether. Borrowing interface components and toolkits from existing platforms might save you time, get you to working code faster, and make the product seem more familiar to wary users...but it can

also result in inappropriate and unusable interfaces. An app for a smart watch almost certainly shouldn't replicate an app for a tablet. Even if your new kitchen appliance runs a web browser, do you really want that browser to work like Chrome?

Indeed, you may need to develop an interaction paradigm from scratch. Are you integrating mechanical inputs? What about gesture recognition or computer vision? How does navigation work? Are there battery, memory, and processing limitations in the hardware to take into account? Are there display limitations, as with a monochrome screen for an ultra-low-power wearable? Can you change the display specifications to support richer screen interactions, or are they fixed?

Developing for new platforms can be slow and expensive, so you'll probably want to do more upfront paper prototyping (see Figure 14-21 for an example), semi-interactive click-throughs, and simulations on existing consumer electronics before moving to code.

### **Evaluating placement, size, and scale**

Thumbnail sketches are great for generating lots of ideas quickly, but there's no substitute for continuing paper prototyping in place, at 100% scale. We still cringe at the memory of discovering that left-handed users could not view the screen of a hand-carried device only *after* building an expensive working prototype.

For small displays, prototyping at 100% scale will help you discover and solve ergonomic problems of vision and interaction—particularly if you attach the paper screen prototype to a rough device mockup. Try role-playing a wearable with a paper screen: what do you have to do to see the screen? How would you feel if strangers could view it? Or, if designing an appliance, attach the small screen to a full-size foamcore mockup of the large object. Can people with nonstandard (in a wheelchair, say) and standard physical capacities use it?

For displays much larger than a typical computer screen, paper prototyping at 100% scale helps you discover and solve similar ergonomic problems (Are key controls too high up for the majority of people to reach? What happens when multiple people crowd around an interactive table-top?), as well as social questions of multiperson viewing and interaction. Taping a full-size paper mockup to a table or a wall is the



quickest way we know to investigate the *feel* of living with a large-scale display. You can also use a projected image, of course—but paper is more flexible and can be easily annotated.

Stimulant, a consultancy that creates interactive environments, writes:

We are usually asked to make ergonomic recommendations for our fabrication and construction partners, so a physical mockup also helps to adjust the angle of interaction surfaces. In our guerilla usability testing, we found that most users felt more inclined to interact with the touchscreen if it were angled more like a drafting table than if it were more perpendicular to the floor. That's the kind of news we can use.<sup>9</sup>

Placement also matters. If you are designing one of those smart fridges, how high up should you place your screen? What about buttons and other controls? Taping a tablet to an actual, full-size refrigerator is a quick way to start asking and answering those questions. Or, as in Figure 14-15, you might use easy-to-move cardboard boxes to prototype appliances and furniture.

## How Do You Decide What to Prototype?

Prototyping connected products and services can take more time and more skill than even experienced industrial and interaction designers expect. One of the challenges for iterating while designing connected devices and services is staying simple and efficient when tempted to make your prototypes complicated and slow. The following sections detail three rules of thumb to keep the iterative cycle going. They synthesize both rigorous scholarship and plenty of painful personal experience.

### **Prototype what you care about**

This is the most important rule. Prototypes, as we pointed out earlier, can't instantiate every aspect of a problem or solution. And some aspects of connected products, such as working hardware or real-time data feeds, may be spectacularly difficult to prototype—or perhaps even impossible.

---

9 Stimulant, "Visualizing BioOncology: Design Behind the Scenes," August 18, 2014, <http://stimulant.com/visualizing-biooncology/>.



**FIGURE 14-21**

Usability testing with a full-size, taped-together paper prototype of an interactive display (image: Stimulant)

Given that time, money, and attention are all limited, you should always use the technique that can answer your *most pressing* questions while investing *the least amount of effort*. If you don't know which questions are most pressing, making a prototype using as little effort as possible can spark those discussions. In this way, prototyping can also trigger necessary process and project decision making.

For connected products and services, answering the most pressing questions generally means prototyping entire systems *before* visual design and industrial design. And it means investigating activities and experiences across multiple components of the service (including

paper and other nondigital ones) *before* specific screens. Industrial design and visual design should also be iterative, but they, as Sifteo and BRCK discovered, have rather more rigid timelines. Chapter 7 examines industrial design in more detail, including design for manufacturing and assembly.

Focusing on functionality doesn't mean that visual and industrial designers don't need to participate in early concept generation and evaluation. They do. Connected products cannot be designed solely by industrial designers, or by digital UX designers. Collaborative work doesn't just help build consensus; it also allows members of different disciplines to learn from one another and benefit from the skills and perspectives of team members they might not have worked with otherwise.

This first rule leads to some others. Let's look at those next.

### **Build as little from scratch as possible**

We understand that many people really enjoy building working prototypes from scratch, in the same way that many people enjoy making their own pastry dough or tailoring their own clothes. Spending a lot of time building a prototype from scratch is undeniably satisfying. However, it is also inefficient. The point of early prototypes is to explore different aspects of the design space and inspire new responses. The point of later prototypes is to refine, evaluate, and prioritize those responses to that space. (Different timelines apply to software and hardware engineering for production, of course.) In the best case, spending too much time and effort on any one prototype slows down your iterations. In the worst case, you can get too attached to a specific prototype, and resist abandoning it even when shown evidence that your audience doesn't find it useful or usable.

This lesson applies to all methods of prototyping, not just digital tools. It's as easy to spend too much time on an elaborate paper prototype as it is soldering wires.

### **Fit the medium to the purpose**

Elaborate or too-perfect styling creates many of the same problems of attachment and time use for prototyping as building too much from scratch. As well, early obsession with prototype aesthetics—particularly while developing a new or emerging product category—can distract from a crucial question: what might the product do?

As Bill Buxton writes in *Sketching User Experiences*, both “high fidelity” and “low fidelity” in terms of functionality and styling can be unhelpful terms for prototyping. Instead, as IDEO partner Duane Bray says:

What’s most important for a prototype is to choose the right fidelity for the question you’re seeking to answer.<sup>10</sup>

The right medium and style are the ones that direct attention to the aspect in question while signaling that other aspects are less important *at this moment*. Of course, those aspects will change over the course of the project as you settle basic “make the right thing” questions and move on to “make the thing right” questions of usability, features, and style.

It’s not that the aesthetics of prototypes aren’t important. They are. But polishing the appearance of early prototypes is generally a poor use of your time and the time of evaluators. Prototypes can have their own aesthetics. Buxton argues that “hand-drawn” screen elements can signal that the interface is in a rough, early state, with hardware prototypes just as rough and “sketchy” (as in Figure 14-3). Alternatively, a deliberately neutral style can signal, as with BERG’s minimal case design, that the industrial design isn’t what’s being examined (Figure 14-4).

Another familiar distinction is that between “looks like” and “works like” prototypes. To this, we and many other people add a third category: “feels like” prototypes. If you’re interested in product experience, you might not want to start by prototyping the *form* of the product at all. Instead, you might be more interested in how an experience *feels*. In that case, you don’t need to build an object that “works” by any definition. Experience prototyping and Wizard-of-Oz demonstrations might actually be more faithful “feels like” prototypes than a barely functional object or two-dimensional storyboards.

---

10 D. Bray, “How I Work: IDEO’s Duane Bray On Creating Great Digital Experiences,” <http://www.smashingmagazine.com/2013/05/04/how-i-work-ideos-duane-bray-interview/>.

## Evaluation: Success Demands Failure

Many companies that loudly proclaim the need for human-centered design, iterative design, or “failing early” often don’t perform the sort of frequent evaluations that they themselves advocate. They walk through prototypes within the team, but don’t invite in outsiders, much less user representatives. Ask them why, and you’ll hear lots of different rationalizations. Sometimes “research restricts creativity.” Sometimes “research is too expensive.” Sometimes “there’s no time.” Sometimes “we want to wait until we do can do it *right*.” Sometimes, “we don’t need to do research because we’re designing for people like us.”

The irony of iterative design is that later success demands early failure. Putting off UX evaluation activities until you have a fully working prototype is the worst possible thing to do when time is short and you have little money. High stakes and infrequent testing with polished, high-fidelity prototypes is an excellent way to doom your product. Using many different types of prototypes to ask questions as you go will more likely produce desirable, useful, and usable products than a single “evaluation phase” after you have a functional prototype. We know that having outsiders criticize cherished ideas can *hurt*. But the results of basing design decisions on guesswork will hurt more.

The choice you face is not between iterative and noniterative design. The choice is between iteration and a refusal to improve.

Planning for iteration is the wisest way to develop any kind of product, much less products as complex as connected products and services. How else are you going to avoid wasting scarce resources? How else are you going to base your directions on observation as well as intuition and guesswork? How else are you going to discover surprising new opportunities you couldn’t have imagined on your own?

One way to make evaluation more frequent is to make it more informal. The word “evaluation” itself can make research seem nerve-wracking—as if a schoolteacher is grading a final paper. At some point, you may want to get serious about ergonomics and usability. However, especially early on, *some research is better than none*.

If the word “evaluation” seems too formal, try calling those activities “learning.” Or “trying stuff out.” Or “customer support.” The BRCK team offers in-person configuration assistance at their studio in

Nairobi. The main goal is to support new customers—but the BRCK helpers are also using the encounters to evaluate the latest changes to the setup process.

Try picking an easy-to-track success metric that doesn't require much calculation. For Sifteo, that metric was: Do fewer than 2 people out of 10 complain about the battery insertion mechanism? Every week, the Sifteo researcher had 10 to 15 people attempt the setup process with revised setup instructions until the complaints fell below that threshold.

In the long run, though, what's important is that learning continues *after* the product launches. As Juliana Rotich says of the BRCK's hardware, software, and instructions, "We're learning and iterating and changing every single day."

The choice and order of prototyping and investigation activities will be specific to each project, based on your goals, your challenges, and your skills. The process should be exciting, not agonizing. Prototype early. Iterate often. Learn well.

## Summary

Prototypes are *experiments* made to answer specific questions. Prototyping starts before a product concept is chosen, and continues long after launch. However, prototyping connected products can be technically complicated. Decide what questions you want to answer before making a prototype will keep you from wasting time and effort.

Early stage concepts often ask us to imagine everyday life with a new product. *Story collages*, *media from the future*, and *video prototypes* make the product feel real, and contextualize it within a service ecosystem, while signaling that the concept is still open for debate.

Experience prototypes, including Wizard-of-Oz demonstrations, can be a lightweight way to explore new features or functionality. Using role play with relatively low-tech props, they evoke the *feeling* of using a product, rather than how it looks or works.

Connected objects become meaningful as components of services. To prototype the relationship among service components, the key is staying at a high level to avoid getting stuck in the details of any individual component. *Business Origami* uses paper cut-outs to map out those relationships iteratively and quickly.

Crafting the details of interactions doesn't necessarily demand working prototypes. *Adapting existing devices, semi-interactive click-throughs, and full-size paper mockups* are all good ways to test and refine imagined interaction modes before committing to hardware. If you are prototyping hardware, however, look for distributors and toolkits with easy-to-use documentation and thriving online communities.

Data-driven UX presents its own challenges, such as prototyping sensing, sharing, modeling, and responding in real time. *Canned data* and Wizard-of-Oz demonstrations help answer questions about the felt *experience* of streaming data and algorithmically driven interactions.

Nevertheless, you will never be able to prototype every aspect of your product. The central recommendation of this chapter: *prototype what you care about*. Making and evaluating many cheap prototypes early on with potential users and other stakeholders will help you avoid more problems and discover more opportunities than waiting to evaluate a single high-fidelity one.

I WANT TO...	TRY...
Decide early on whether an audience sees value in an idea for a new product or service	<p><i>Video sketching</i> to demonstrate convincingly how the product might work</p> <p><i>Story collages</i> to set conversations about future technologies in a believably "everyday" context</p> <p><i>Wizard-of-Oz prototypes</i> to dramatize nonworking technology quickly and cheaply</p> <p><i>Media from the future</i> to imagine how outside observers would explain the value of your idea to them</p> <p><i>Experience prototyping</i> to roleplay everyday encounters with the product</p>
Identify product/service components and the relationships among them	<i>Business Origami</i> to map relationships among components iteratively, collaboratively, and tangibly
Investigate interactions with specific service components	<p><i>Paper prototyping</i> at 100% scale to address ergonomic factors</p> <p><i>Wizard-of-Oz prototypes</i> to quickly communicate data-driven interactions</p> <p>Theatrical <i>experience prototyping</i> to invoke in-the-moment emotions</p>





# Designing Complex, Interconnected Products and Services

**BY CLAIRE ROWLAND<sup>1</sup>**

Most of the consumer IoT products on the market right now offer a limited set of functions through a small set of devices. As we saw in Chapter 10, many don't interoperate well or at all with third-party devices. This keeps things (relatively) simple in conceptual terms. But IoT is called the Internet of Things because our ambitions stretch further than this, to create interconnected networks of many devices, web services, and users. Your product may grow to interoperate with other third-party devices or services, or become a *platform* supporting many different devices and services. As it develops, the pattern of logical interconnections between devices, data, and users will become very complicated, very quickly.

Technical interoperability is a huge challenge in its own right. But creating a design that allows users to understand, anticipate, and control how interconnected services and devices work together is an equally big challenge. The makers of IoT technology are only just beginning to address this.

This chapter looks ahead to a future of complex systems composed of many interconnected devices and applications. It discusses the challenges in making complex systems understandable and valuable, keeping users in control without overwhelming them with configuration options.

---

<sup>1</sup> I am indebted to Alan Blackwell for the observations and pointers to research that inspired the discussion of programming-like experiences.

This chapter introduces:

- How systems become complex as they begin to scale (see page 612)
- The UX challenges of scaling the system (see page 617)
- The challenges of putting users in control of complex systems (see page 624)
- The challenges of handing over control to the machines (see page 646)
- Approaches to scaling UX in the platform (see page 653)
- Using data models to coordinate devices and systems around user needs (see page 656)

This chapter addresses the following issues:

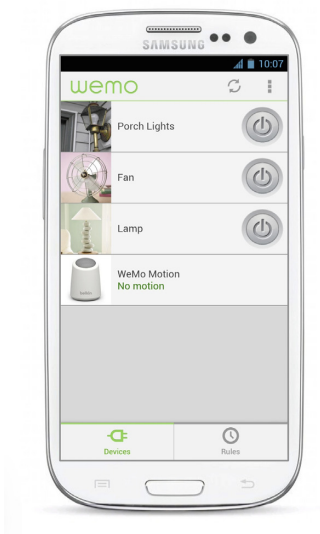
- How adding just a few more devices to a system can create a complex web of interrelationships for the user to navigate (see page 615)
- Why adding more devices to a system means the UX focus has to shift to activities, instead of controlling devices (see page 620)
- Why putting users in control of IoT is asking them to think like programmers, and why that's a challenge (see page 626)
- Why handing over control to machines can make complex tasks easier... as long as we don't lose control (see page 646)
- Using data models to describe devices, system components, the context of use and the user (see page 656)

## It's Complicated...

In Chapter 9, we saw that even fairly simple connected devices and services can be conceptually much more complex than nonconnected equivalents. Users need some grasp of how functionality is distributed around the system, and how it may fail if parts go offline.

But so far, we've talked about examples that are relatively straightforward. Most of the consumer IoT systems out there involve a small set of known devices. Many are designed for only one user, or are blind to the fact that they may have multiple users. As we saw in Chapter 10, many of these are controlled via a software platform (and often web/mobile

apps) provided by the manufacturer. The UX design of these apps is often geared toward controlling the individual devices—for example, Philips Hue or WeMo (see Figure 15-1).



**FIGURE 15-1**  
Device controls in the WeMo app (image: Belkin)

When you only have a few devices, this is perfectly acceptable. You open one mobile app to control your thermostat, another to control a couple of lights, and perhaps one more for a connected socket or two.

But once you start adding more and more devices, things become complicated. As discussed in Chapter 10, you could soon end up with lots of apps, each controlling one or two devices. Just finding the right app for the thing you want to do becomes a challenge. One app giving you individual controls for lots of devices isn't necessarily much better if you have to dig deep to find the right device or control.

But making device controls easy to access is only a part of the problem. As you add more and more devices, the number of potential interconnections between them increases rapidly.<sup>2</sup> To make those connec-

---

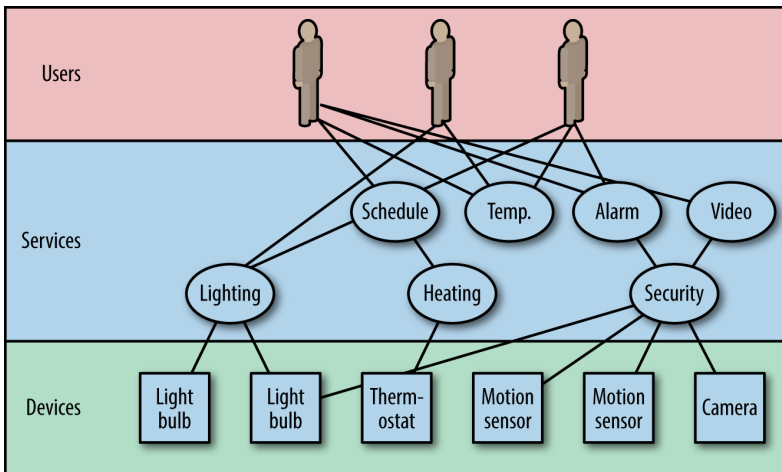
2 The mathematical formula underpinning Metcalfe's Law states that the number of unique connections in a network of  $n$  nodes is  $n(n-1)/2$ . So two devices can only form one connection, but five can make 10 connections and 12 can make 66 possible connections. See [http://en.wikipedia.org/wiki/Metcalfe's\\_law](http://en.wikipedia.org/wiki/Metcalfe's_law).

tions meaningful, we'll want to add more and more ways for them to coordinate with each other. That creates more potential interrelationships between devices, and more device and cross-device functionality to understand, configure, and control. The most complex consumer example of this is currently the connected home.

The whole point of IoT is that the sum of the parts is much more powerful than the individual components. Considering automation and interactions between device types is vital when designing easy to use and flexible products.

**GAWAIN EDWARDS, IOT CONSULTANT<sup>3</sup>**

The example in Figure 15-2 represents a fairly simple connected home setup.



**FIGURE 15-2**

Interrelationships between devices, services, and users in a simple connected home setup

There are six devices: two connected lights, a thermostat, two motion sensors, and a security camera.

There are three user-facing applications (or services): security, heating, and lighting.

<sup>3</sup> Personal communication.

Some devices support more than one service, so for example, motion sensors are part of the security system but also provide temperature data to the heating system. And one of the lights can be activated by the security alarm to provide illumination for the camera.

Three users are currently set up on the system. The system allows different levels of permissions to be granted to different users on different services. The “admin user” can access all the functionality, but the other two (who might be regular visitors or other household members) have limited access. They can adjust the heating temperature but not set the schedule. Only one can activate the security alarm, and neither can view the video feed from the camera.

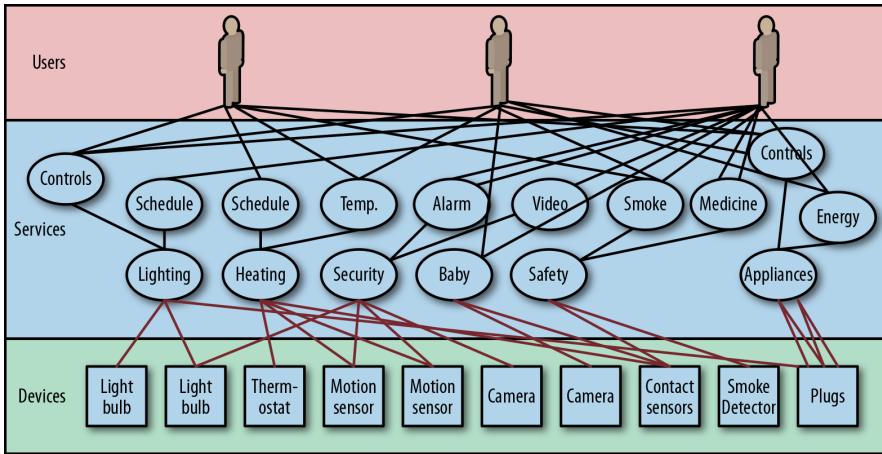
Although this is a fairly simple system, there is already a complex pattern of interdependencies and relationships emerging between the devices, services, and users. The more devices and services are added, the more complex this will become.

Here’s the same system with just nine extra devices added, plus additional lighting controls (in Figure 15-3):

- An extra camera in a baby’s room (accessed via a baby monitoring service).
- Three contact sensors that also measure temperature: one on a medicine cupboard (part of a safety service), one on the baby’s window (also feeding temperature data to the baby monitoring service), and one on another door. All of these also supply temperature data to the heating service.
- Four smart plugs: all accessed via an appliances app with energy monitoring and controls. One that is on a lamp is also available to the lighting service.
- A smoke alarm, accessed via the safety service along with the medicine cupboard alarm.

As in the previous diagram, not all users can access all functionality.

This is already a complicated web of interrelationships, but in device and service terms it’s still pretty basic. Most of us have more than 2–3 lights, and even a medium-sized house will need more than 4–5 alarm sensors. Effective energy monitoring would require a whole house reading (e.g., from a smart meter).



**FIGURE 15-3**

Interrelationships between devices, services, and users in a slightly more complicated (but still fairly basic!) connected home setup

And there are additional ways of controlling and organizing devices and services that aren't shown here. Many of these devices may be assigned to rooms in the house. Functions from various services may be controlled by automated rules, which are user-configured groups of behaviors. Some of these may be dependent on tracking the presence of the various occupants and visitors. A typical rule might be "if it's after sunset and no one is home, turn on the living room and kitchen lights," to make the house look occupied.

In a system like this, enabling the user to understand the system and keep track of everything that's going on is a major design challenge. And we're not even taking account here of the challenges of interoperability (as explored in Chapter 10), or interusability between the various control interfaces (as explored in Chapter 9).

Understanding the interrelationships and interdependencies becomes yet more complicated when the system (or just some devices) is/are autonomous or intelligent in some way. Instead of responding only to direct user commands, the system infers what users need and acts independently. Done well, this is powerful: users want devices to behave in common sense ways, which requires intelligence. But users can also fear loss of control. The smart home/environment gone wrong is a

well-known trope in contemporary fiction, from HAL in the film *2001: A Space Odyssey* to the sinister HappyLife Home in the Ray Bradbury story “The Veldt”.<sup>4</sup>

When domestic technology goes awry, it is often more invasive than office technology; not only do we expect our homes to provide a haven of calm and security, but breakdowns in domestic technology can actually prevent us from meeting our basic needs.

**JENNIFER RODE, ELEANOR TOYE, AND ALAN BLACKWELL**<sup>5</sup>

Making sense of this complexity is an enormous challenge for UX. As yet there are no easy answers, but unless there is progress in this area, a consumer Internet of Things will remain unrealizable.

## Scaling the UX

In general, the more devices are added to a system, the more complex it becomes to accommodate them in a UX design that works well for users.

Scaling the UX is a particular issue for systems that support lots of actuators and user-facing controls, because each new device added is potentially increasing the amount of user-facing functionality required.

It’s less of an issue for networks of sensors, whose UX is likely to be data driven. Adding new data points may require you to change the way you present insights to the user, or enable new insights. But adding a new set of controls creates more interdependencies for the user to understand.

### KEEPING TRACK OF MULTIPLE DEVICES

Handling multiple devices raises practical issues. Just identifying devices can be a challenge if you have several of the same type, such as multiple motion sensors. Is the device in front of you “motion sensor

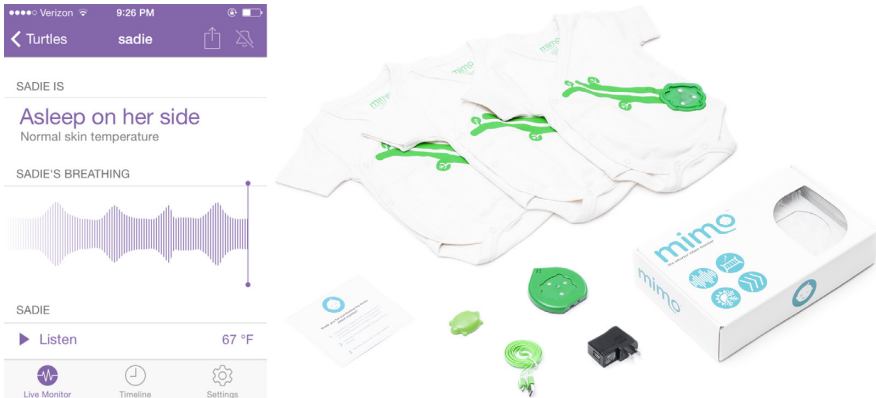
---

4 Ray Bradbury, “The Veldt,” in the 1951 anthology *The Illustrated Man* (and here: [http://www.d.umn.edu/~csigler/PDF%20files/bradbury\\_veldt.pdf](http://www.d.umn.edu/~csigler/PDF%20files/bradbury_veldt.pdf)).

5 Jennifer Rode, Eleanor Toye, Alan Blackwell, “The Fuzzy Felt Ethnography—Understanding the Programming Patterns of Domestic Appliances,” *Journal of Personal and Ubiquitous Computing*, 8:3-4 (2004), <https://www.cl.cam.ac.uk/~afb21/publications/2ADfuzzyFelt.pdf>.

1,” or “motion sensor 2”? The system needs to differentiate them, and the user needs a way to differentiate them in the interface, such as giving them names. And there also needs to be some way of matching the physical device in front of you with the correct representation named in the interface. As we saw in Chapter 12, some systems provide an “identify” button on the UI that makes an LED flash on the device.

Keeping track of which device is broadcasting what information can be tricky. For instance, say you have twins and two baby monitoring sensor kits. Do you want each transmitter always to inform you about the well-being of the same baby (this might save valuable seconds in an emergency)? The Mimo baby monitoring kimonos from Rest Devices (see Figure 15-4) monitor baby vital signs, like sleep position and breathing. A standard kit for one child comprises three kimonos with onboard sensors, one Bluetooth transmitter (called a “turtle”) that fits onto the kimonos, and one gateway (called a “lilypad”). Parents of twins can use one lilypad but need a turtle for each baby. If you want to keep the data from each baby separate, you must always use the same turtle for each child.



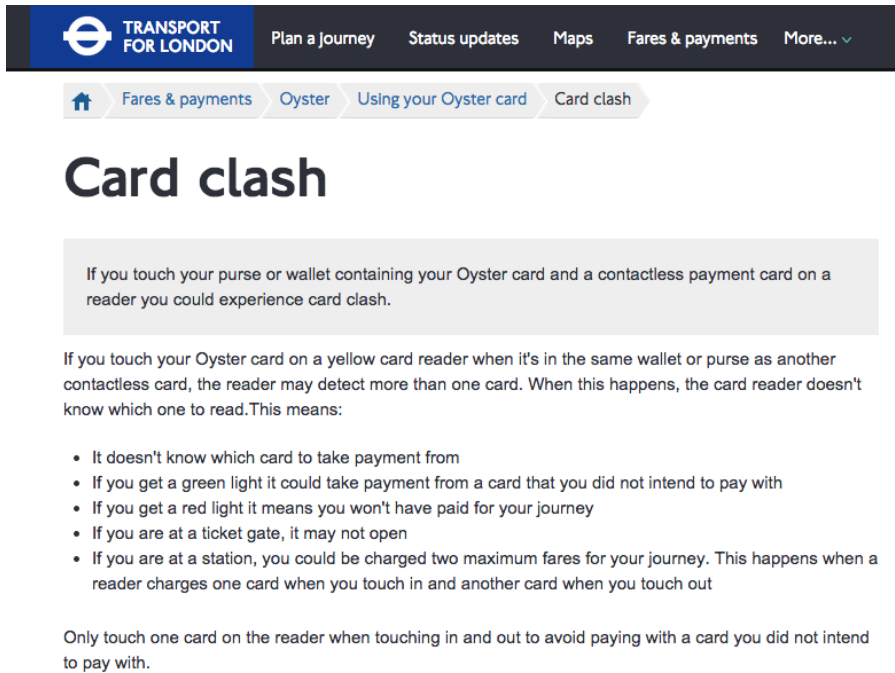
**FIGURE 15-4**  
The Mimo baby kit (image: Rest Devices)<sup>6</sup>

With multiple devices performing similar functions, there is the possibility of choosing the wrong one. The introduction of contactless credit cards as an alternative to the Oyster card has created the risk of “card

<sup>6</sup> <http://mimobaby.com/>



clash” where two Oyster-compatible payment cards are kept together in a wallet (see Figure 15-5). The user could inadvertently swipe in on one at the start of the journey, and swipe out on the other at the end, incurring two penalty charges for “incomplete” journeys instead of the normal fare. It would make sense to link both cards to the user’s account and combine the data from the two, but this isn’t possible on the current system.



**FIGURE 15-5**

The Transport for London website warns of the risk of card clash (image: TfL.gov.uk)

Identifying devices isn’t the only challenge, either. If you have lots of devices, just configuring them to work together is a huge overhead.

Keeping track of what they are doing and predicting how they may behave can be challenging, too. In a complex system, that might mean knowing which one is connected to what, what each is doing (right now), which software or which user is controlling it, what data it creates, and who can see it.

Adding automated behaviors into the mix makes things yet more complicated. Some devices may be triggered not only by explicit user interactions, but by preconfigured rules established sometime in the past, or perhaps system intelligence. So there are more interdependencies to map, and it becomes harder to understand what the system is up to and why. The conceptual model is far more complex.

## **LOOKING BEYOND DEVICES**

As already discussed, device-centric UXes are fine for small numbers of devices. It will be quite easy to find the controls you need by looking for the specific device. You don't need to create semantic groupings (e.g., by room or function), to find the one you need.

But if you have 100 devices, a device-centric UX becomes more burdensome. Trawling through 99 other devices to find the one you want, like a living room light, could take a long time. You don't even have to get as far as 100 devices to get fed up with controlling them one at a time: you may only have 3 lights in your living room. but it's quite reasonable to want to turn them all on at once. In that case, you might want to group devices, perhaps by their location in the house, or by function. But you might still want to turn just the one reading lamp on or off, so you'll still need the individual device control. And you might want to put a device in two overlapping groups: perhaps one for "living room" and one for "security lights."

The more complex the system, the more you have to design it around what it does for the user and not the devices. For example, users don't want to have to monitor lots of sensors just to know they've left a window open or someone has broken in. As your system scales, it will not (always) make sense for the UX to focus on individual devices anymore.

As with many of the examples in this book, we'll look at the connected home as an example, as it's the environment in which the consumer is likely to have the most devices working, possibly together. Homes are challenging to design for, as they are complex social contexts, which we view as our refuge from the world. They are the last place we want to be frustrated in our daily routines by bad design or unreliable technology, and within the home (as opposed to, say, the workplace) we are mostly able to choose the technology we use. So the bar for perceived value and good UX is set high.

Some home devices, like thermostats or the garage door opener, might be “first order” devices that map closely to a user need. These may be under the user’s direct control and represented prominently in the UI. Others, like contact sensors, may not be that meaningful as standalone devices. They are more valuable as part of a service, like the security alarm. Some devices, like lights, could exist in both categories: you want individual control of a light, but also to use it as part of a group.

The more devices your system has, the more impractical it becomes to rely on device-level controls and the more the UX needs to shift to higher-level service or application behaviors. Services might include irrigation, heating/cooling, medicine cabinet alarm, smoke alarm, energy monitoring, window shades, and Granny’s emergency alarm. A service might have direct device controls (e.g., the heating/cooling service includes thermostat controls, but also home-wide temperature monitoring), or like the intruder alarm, it might derive its function and value from a network of devices. It is application software that determines whether the security alarm is “set,” “unset,” or triggered, not any of the individual devices.

Users need you to be considerate in managing their attention, and a good UX will organize functionality in ways that map well to their higher-level needs and goals. As we saw in Chapter 5, these might be very varied, inconsistent, and driven by the complex contexts in which the devices are used. In the next section, we’ll look at some possible ways of organizing functionality to suit daily life in the home.

## **ADDRESSING USER NEEDS**

Let’s consider some of the needs a user might have around the home and what that might mean for organizing devices and functionality in the system. There are multiple, overlapping ways of making sense of this (see Table 15-1).

Some can be viewed in terms of *devices*: tightly mapped to a single device, like turning off the TV, or a group of devices, like the master switch that turns off all the lights.

Some might target a particular *location*. Many connected home systems allow users to group devices by room. For example, the living room group might give you one easy place to control your immediate environment without leaving the sofa. But not all user needs fall easily into room groups. Heating and cooling may have room-based data or controls, but usually cover the entire house. The concept of location might extend to turning off the alarm *downstairs*, turning off the *outside* lights, or lowering the blinds on *the west side* of the house. Classifying every single device by all the potential ways you could describe its location would easily become overwhelming.

Location is a useful way to describe devices like fridges and televisions that don't move around often. Smaller devices, like lamps, or smart plugs, may get moved. There may also be several in a room, making location information less useful in identifying them. Some devices, like baby monitors, are designed to be carried around and the user may forget where they are.

Sometimes user goals might be *function based*, such as setting the hot water to come on at 5 p.m., or monitoring safety alarms. Or they could be *activity based*, such as getting ready in the morning (needing hot water, coffee, and transport updates), or watching a movie (turning down the lights and turning on the TV and surround sound system). Connected home systems often offer the option to group functionality as “scenes” to support different activities. For example, there might be a “dinner party” lighting scene, a “movie scene” that turns off most lamps and turns on the home entertainment system, or a “goodnight scene” that turns nonessential devices off when everyone goes to bed.

Needs could also be *person based*, such as notifying you if Grandpa hasn't got out of bed, or letting you know when Jake gets home from school. Or they might be *authorization based*, perhaps locking the medicine or liquor cabinet when there are no adults at home.

Sometimes needs are *time or state based* (e.g., activating security lights at dusk, and/or when the house is unoccupied). Some may be *optimization based* (e.g., keep the house temperature comfortable, use energy efficiently). This is a tricky need to fulfill, as defining “comfortable” or “efficient” requires information about the current state and the wider context of use.

**TABLE 15-1.** A taxonomy of user needs (needs can fall into multiple categories)

TYPE OF NEED	EXAMPLE
Device/group of devices	<ul style="list-style-type: none"><li>• Turn on the TV</li><li>• Turn off all the lights</li></ul>
Location	<ul style="list-style-type: none"><li>• Set the alarm downstairs</li><li>• Turn off outside lights</li><li>• Lower the blinds on the west side of the house</li></ul>
Function	<ul style="list-style-type: none"><li>• Turn on the hot water</li><li>• Dry the laundry</li></ul>
Activity	<ul style="list-style-type: none"><li>• Help me get ready and out of the door on time in the morning</li><li>• Set the living room up to watch a movie</li></ul>
Person	<ul style="list-style-type: none"><li>• Tell me if Grandpa hasn't got out of bed</li><li>• Tell me when Jake gets home from school</li><li>• Track my weight and blood pressure</li></ul>
Authorization	<ul style="list-style-type: none"><li>• Lock the medicine cabinet when the adults are not at home</li><li>• Let the courier into the porch</li></ul>
Time	<ul style="list-style-type: none"><li>• Warm the house for 7 a.m.</li><li>• Don't run the sprinklers between 11 a.m. and 7 p.m.</li></ul>
State	<ul style="list-style-type: none"><li>• Turn off nonessential devices when I'm on vacation</li><li>• Turn on the security lights at dusk</li><li>• Disable notifications when I'm driving</li></ul>
Optimization	<ul style="list-style-type: none"><li>• Keep the house temperature comfortable</li><li>• Use energy efficiently</li></ul>

## CONTEXT

Creating a UX that bridges this collection of user needs with a bunch of devices requires some understanding of the context in which the devices and the system are being used. The range of contextual information you could capture is huge, but physical, activity, and social contexts are all good categories to consider (see Chapter 5).

The physical context of a device may affect how it works. For example, your window alarm sensors may well be able to measure temperature. That might seem like useful data to help control your heating system, too. But they are likely to read a lower temperature than the center of

the room or an internal wall, which better reflect the temperature as perceived by a human in the room. Using them to control the heating or AC in that room may result in it feeling too warm, as the heating overcompensates (or AC undercompensates). Understanding the physical environment of the device will allow you to interpret and use its data more accurately.

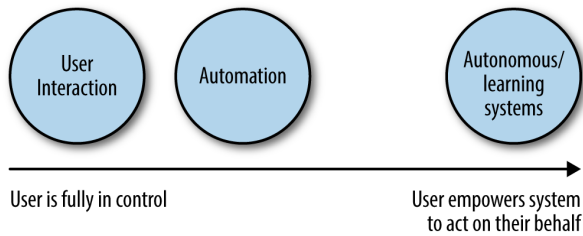
The extent to which the system supports or conflicts with the activities that happen in the household can determine how effective or valuable it is. It's easy, for example, to wish to use energy "efficiently" and "reduce waste" (as mentioned earlier) but quite hard to translate that into actual rules as to which appliances the system should turn on/off and when. It's not just about using less energy, which would mean turning everything off. That requires an understanding of how the household weighs cost and efficiency savings over comfort and convenience, and which appliances or energy needs are most important (e.g., cooling the baby's room, having a clean, dry shirt by 8 a.m.).

As discussed in Chapter 5, homes are complex social contexts. You may need to consider people who live in the house, have access to enter the physical property, or the service UI. You may have to capture contextual information such as presence (who's in or out), who is authorized or not authorized to access particular functionality, proximity (who is nearby or far away), and who is available or unavailable. Proximity and availability might be used to identify the most suitable respondent if the security alarm goes off—who is nearby, and likely to receive the message/be able to get home?

You don't necessarily want to model all the relationships between different stakeholders in your system, but you should consider how your design supports or conflicts with the social context of the home as you create it. For example, a really rigid permission system might seem to offer lots of control but will force users to do lots of categorizing of who is allowed to do what. This may suit law-and-order types with time on their hands, but won't suit those who like to be more flexible sometimes (perhaps allowing little Jimmy to stay up late to watch the World Cup). And it could cause friction.

## Control

As systems become more complex, two contrasting approaches to controlling them have to be weighed up (see Figure 15-6).



**FIGURE 15-6**

Approaches to controlling IoT systems

On the one hand, we can try to create intuitive interfaces for users to control the systems themselves. This is where many consumer IoT products are right now: remote controls are often the core of the value proposition. The next natural step, also user controlled, is automation: the user configures functionality to run automatically in the future (e.g., by setting up heating to come on at 4 p.m.).

On the other, we can hand over control to intelligent or autonomous systems, empowered to act independently on our behalf (e.g., the heating system infers that it should come on at 4 p.m. because you are likely to be home).

Both approaches have advantages and disadvantages.

People are good at intuiting context, especially social context and “common sense” (unlike machines). Remote controls make sense, until there are too many of them to manage. But complex systems require us to understand interrelationships. And as we saw in Chapter 9, people may not be very good at understanding networks. What’s more, people are notoriously bad at predicting their own future needs, so automation can be problematic (this is discussed in depth later). People also get tired, bored, distracted, or overwhelmed by other demands on their time, and sometimes behave irrationally and against their own self-interest (e.g., using thermostats in inefficient ways).

Computers are good at storing and analyzing huge amounts of information, logical reasoning, and maintaining constant vigilance without getting tired or distracted. But computers are terrible at understanding context (unlike people). Teaching a computer the kind of “common sense” that humans take for granted is a very challenging artificial intelligence problem. The computer has to be taught what information is relevant to the situation and how to interpret it.

Of course, this isn't an either/or choice. It often makes sense to combine some user-facing controls with system intelligence. In the next section, we'll look at some of the challenges of each approach individually.

## **PUTTING USERS IN CONTROL**

In this section, we'll consider the design challenges of putting users in control of complex systems, and why using IoT is a lot like programming.

### **Why is using IoT like programming?**

Back in 2002, Ann Light suggested that “we stop thinking about products' end-users and start thinking about the system's end-designers. Because that is what we are all becoming as we choose and use network components.”<sup>7</sup>

For the past 30+ years, the prevailing trend in UX has been toward direct manipulation: “a human–computer interaction style which involves continuous representation of objects of interest and rapid, reversible, and incremental actions and feedback.”<sup>8</sup> This is the approach behind tools such as WYSIWYG (What You See Is What You Get) editing in word processors and web production tools like Dreamweaver. It's also evident in the new generation of content-driven UIs found on modern smartphone and tablet UIs such as Windows Phone 8 and iOS 7 (see Figure 15-7), and natural user interfaces (NUIs) like the Kinect (see Figure 15-8).

Direct manipulation interfaces tend to be easier to use as objects and possible actions are laid out in front of the user. This lessens the effort required to build an abstract mental representation of the scope and capabilities of the system (as you would have to with, for example, a command-line interface). The state of the system is visible: users can immediately see the effect of their actions on the objects in front of them, and choose to reverse them if needed. Users understand what's going on, what actions are available, and need not fear breaking things.

---

7 Anri's Rant: Users as Designers—Challenge of the Network Age UN, May 6, 2002, <http://usabilitynews.bcs.org/content/conWebDoc/47049>.

8 [http://en.wikipedia.org/wiki/Direct\\_manipulation\\_interface](http://en.wikipedia.org/wiki/Direct_manipulation_interface)





**FIGURE 15-7**

Windows Phone 8 on the HTC One M8 (image: Microsoft)



**FIGURE 15-8**

Playing Kinect Sports 2 on the Xbox (image: Doug Kline via Wikicommons/CC license)

A key condition of direct manipulation is that the user's actions and the results happen at the same time, in the same situation.

But consider these interactions:

- Turning your heating off remotely via smartphone.
- A security alarm that is activated by one or more motion or contact sensors being triggered, which sounds an alarm, sends a text message or app alert to the homeowner, and records two minutes of video on the hall camera.
- A heating system that sets the internal target temperature to 21°C when the homeowner's smartphone location is less than one mile from home and heading toward home.
- When Mom or Dad are not at home and the medicine or liquor cabinet is opened, send an alert to Mom and Dad's phones.
- An energy customer agrees to let their supplier company temporarily suspend power to certain appliances, such as the dishwasher, during periods of high demand on the grid. In return, they receive a cheaper tariff.
- A user configures a smart home system with an "away" mode, so that when they leave the house, the heating is turned down, the security alarm is set, nonessential electrical appliances are turned off, and the lights are off (unless it is after dusk and before midnight, in which case a pattern of random security lights is activated).

All of these scenarios break direct manipulation, because they involve the separation of user actions and system responses. IoT services are inherently distributed across multiple devices, and often involve remotely controlling one device from another in a different location, or configuring devices to perform actions at different times or in different conditional situations. This requires the user to expend more effort building and maintaining an abstract mental representation of the system and the consequences an action may produce. It means that using, and especially configuring, an IoT product or service is often more like programming than our general experiences of using UIs. Our actions have become displaced from the results of our actions:

Programming permits us to initiate appliances' actions at future times, or to create macros to make repeated tasks easier.

**JENNIFER RODE, ELEANOR TOYE, ALAN BLACKWELL**<sup>9</sup>

People are often bad at articulating their own current needs. They are even worse at predicting their future needs, and often get them wrong. When your goal is to configure a system to perform actions in the future, this is a barrier. Even more challenging to users is the need to anticipate all the changing conditions to which the system might have to respond.

In the next section, we look at a number of ways in which displacement between actions and results can cause unintended effects.

### **How IoT products displace actions from results**

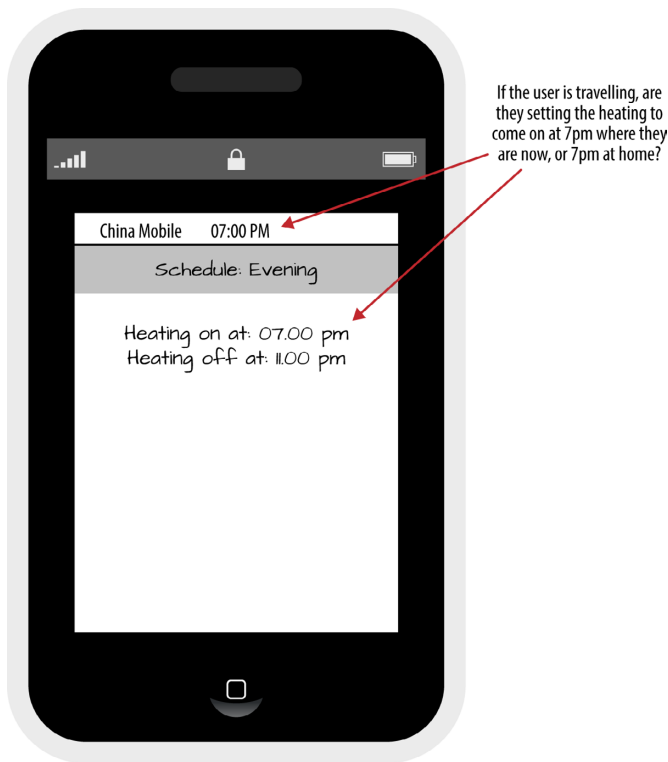
**Displacement in space.** When the action and execution happen at different locations, the user may not be aware of what's going on at the other end and inadvertently cause problems. The action may be inappropriate or unsafe at that time, such as setting the security alarm when there's still someone at home, or more dangerously, turning on an electrical appliance when there's a gas leak in the home. Users may not see the consequences of their actions, which is particularly important if the action was done unintentionally (e.g., opening the garage door by accident). In addition, users may not realize that the action they thought they took didn't work as planned: for example, a user might think they turned off the smart plug on the hair straightener (a common appliance that people often worry they've left on), but didn't, or the command got "lost" by the system. Confirmation as to whether actions have been performed or not is, of course, critical.

Sometimes, the difference in location between commands and actions causes unintended confusion or worse. A user of a remote control heating system once asked us whether, when he was traveling and used his phone to set his home heating to turn off at 7 p.m., the heating would

---

9 Jennifer Rode, Eleanor Toye, Alan Blackwell. "The Fuzzy Felt Ethnography—Understanding the Programming Patterns of Domestic Appliances." *Journal of Personal and Ubiquitous Computing*, Volume 8 Issue 3-4, July 2004. Available from <http://www.cl.cam.ac.uk/~afb21/publications/2ADfuzzyFelt.pdf>.

be turned off at 7 p.m. local time to his house, or 7 p.m. in the time zone where his phone was currently located (see mockup in Figure 15-9).



**FIGURE 15-9**

Mockup of a mobile app showing local time in info bar and schedule setting in app

Both of these scenarios require users to have a fairly sophisticated understanding of the system to predict and explain likely behavior in different locations.

**Displacement in time.** Here, users are effectively configuring a system to execute an action at a future time. An example might be a well-intentioned house rule that turns off “unnecessary” appliances to save electricity overnight. This requires users to anticipate their own future needs, which of course humans are not very good at. That “unnecessary” appliance that gets turned off might be the tumble dryer running overnight to ensure a shirt is ready for an important meeting. This is

also an example of the consequences of users forgetting that they have set up an action, and then being negatively impacted by the execution of that action.

**Displacement in function/application.** Another factor that can make it more complex for users to understand the cause-effect relationship of system actions is multifunctionality, or the repurposing of one device or piece of data for a different or additional use than that originally intended. This requires users to understand additional consequences: for example, an “intelligent” security system could in theory draw on information from external services, like your calendar, to detect possible intrusions and avoid false alarms. Imagine you enter an evening social event into your calendar, which is subsequently cancelled at the last minute. You don’t bother to update your calendar, and head home. On attempting to unlock your front door, the alarm sounds: the system thinks you’re out and so identifies you as an intruder. And which rule dominates? Should it be the rule that turns the lights down when you’re watching a move, or the rule that turns the lights on to welcome someone home?

If a device set up for one usage is subsequently used for a second, it may function less well as a result of being inappropriately configured for the second usage. Some devices, notably smart plugs/outlets, can be moved and repurposed. A smart plug may be placed on a lamp in the living room, controlled by a rule to turn the lamp on for 10 minutes when motion is detected. If the plug is moved to a computer in the study without the system being reconfigured, the rule will now turn the computer on and off, potentially damaging it. Or if the plug is moved to a set of hair straighteners, it could even start a fire. A smart system would detect when the power consumption pattern of the smart plug had changed and ask the user to confirm whether it was now being used on another device, but there’s no reliable way to prevent errors.

Of course, users must, where possible, explicitly grant permission before new uses of data or devices can take place, even if it’s just making a light available to the security alarm. But it’s almost impossible to prevent users from doing ridiculous, unwise, or even dangerous things with a system. There are many potentially unintended consequences of repurposing data and devices for secondary uses, and in a highly

interconnected world, it may be hard for users to unpick these interrelationships and understand why certain things are not working as expected (not to mention what data is held about them and where it may end up).

### **Programming and notations**

As well as the need to form an abstract conceptual model of the system and its capabilities, a user engaged in a programming activity must generally also learn a form of notation in order to create instructions for the system. This is yet another level of abstraction and cognitive overhead. Examples of notation systems that can be used with IoT services include If This Then That and smart home rules. But most of these systems are better suited to regular rules than the flexible permissions that govern many of our home lives—for example, if Jenny gets a good grade on her math test, then she can stay up late playing Xbox this weekend.

### **What is programming?**

At what point does usage or configuration become programming? People argue at length about what is or is not programming—for example, whether using a scripting language such as HTML or Excel macros should be included in this category. For our purposes, this doesn't matter. Our definition of programming represents a particular mindset and set of complex cognitive tasks that are, from an end user perspective, more mentally demanding than those typically required to use a product or UI.

First, users must understand and frame their own problem. Perhaps they want to keep the garden lawn nice and green, whatever the weather. In software engineering terms, this is *requirements definition*. Then, they must figure out which tools they need to solve it (perhaps a connected garden sprinkler and maybe some moisture sensors, see Figure 15-10). This is *specification*.

Next, they must break down the problem into logical components. They have to understand the “normal” operation of the system but also need to anticipate future conditions and needs that might provide exceptions, such as hosepipe bans, the grandkids visiting and playing outside, and barbecues (to avoid the impromptu soaking of house guests). They then need to figure out how to get the tools they have to produce the desired actions: a problem of *software design* involving some form of abstract representation, or *coding*.



**FIGURE 15-10**

The skydrop smart irrigation controller (image: skydrop)

Finally, they have to check it actually works (*testing*), which may involve simulating conditions of use or simply letting it run for a while and seeing whether it behaves as expected. If it soaks the barbecue guests, they'll have to figure out why and fix it—aka *debugging*. And if any part of that system is autonomous or learns over time, some bugs may only reveal themselves after lengthy use.

As illustrated in Figure 15-11, all of these—specification, design, testing, and debugging (and of course maintenance)—are software engineering problems. These can be interesting challenges, but they involve an unusual amount of effort for a consumer product. As discussed in Chapter 5, whether consumer users invest this effort will depend not just on their ability to do so, but also whether they perceive the value of the end result to outweigh the effort or risk of doing so.



**FIGURE 15-11**

The software engineering process

Programming closed systems is complex enough, but interconnectivity adds yet more challenges. Instructions and data may end up being used with devices or services other than the ones for which they were originally intended or authorized, which makes understanding the potential consequences of your actions far more complicated. You might have a home automation rule that turns off heavy power consuming devices

when you go to bed, but if you subsequently buy an electric car that needs charging overnight, you don't want that to lose power, too. Or devices shared between systems may be subject to conflicting instructions: the energy system is trying to turn off nonessential appliances at night when you're on holiday, but the lighting system is trying to turn lights on to make the house appear occupied.

The risk for users is of losing control through sheer complexity. Interconnected autonomous systems, which make decisions and act independently, will be even more unpredictable (see “Autonomous/intelligent systems”). A related issue is retaining control of data and privacy, addressed in Chapter 11.

For every action or system behavior configured, users must consider the following:

- How it will function under currently known conditions (e.g., which appliances should be run after 11 p.m. to make the most of cheap electricity).
- What may happen in the future if some of those conditions change—for example, if the timing of cheap electricity changes. Or, as might happen at Little Kelham in the connected home case study, the user switches on an appliance when solar-generated energy is available and cheap but if everyone else does the same, then the price may rise while they are using it.
- What may happen if the data or behaviors are shared with other apps, devices, tasks, people, or services. Are your well-meaning energy saving rules now preventing your guests from turning on the lights or kettle in the middle of a sleepless night?

Professional programmers do not expect to write bug-free code the first, second, or even third time around. Consumer users are extremely likely to misconfigure automation rules, causing unexpected things to happen. System designers must do everything they can to mitigate any damage.

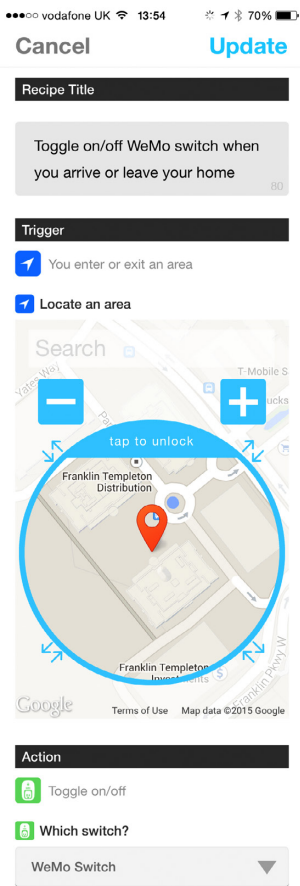
### **How can we make controlling complex systems easier?**

Programming will frequently be too hard, or at least too effortful, for users. What can we do about this? There are three broad solutions here: make programming easier, find an alternative to programming, or get someone else to do it.



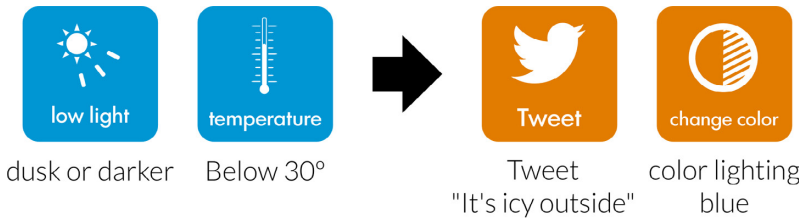
**Make programming easier.** Many attempts have been made to make programming-like activities easier for end users. Most attempt to make programming feel more concrete, generally through simplified graphical approaches.

If This Then That is probably the best known consumer example, which can be used with a number of Internet of Things devices and systems (e.g., Philips Hue, Belkin WeMo products, and SmartThings devices). IFTTT (see Figure 15-12) offers simple *trigger-action* commands: so when a trigger event occurs (such as a child’s smart key fob arriving home), an action can be executed (such as texting the parent that the child is home from school).



**FIGURE 15-12** IFTTT can be used to control a WeMo Switch

Another example of a consumer-oriented programming experience is WigWag: a connected home kit based around sensors and tags designed explicitly to support similar trigger-action programming (see Figure 15-13).



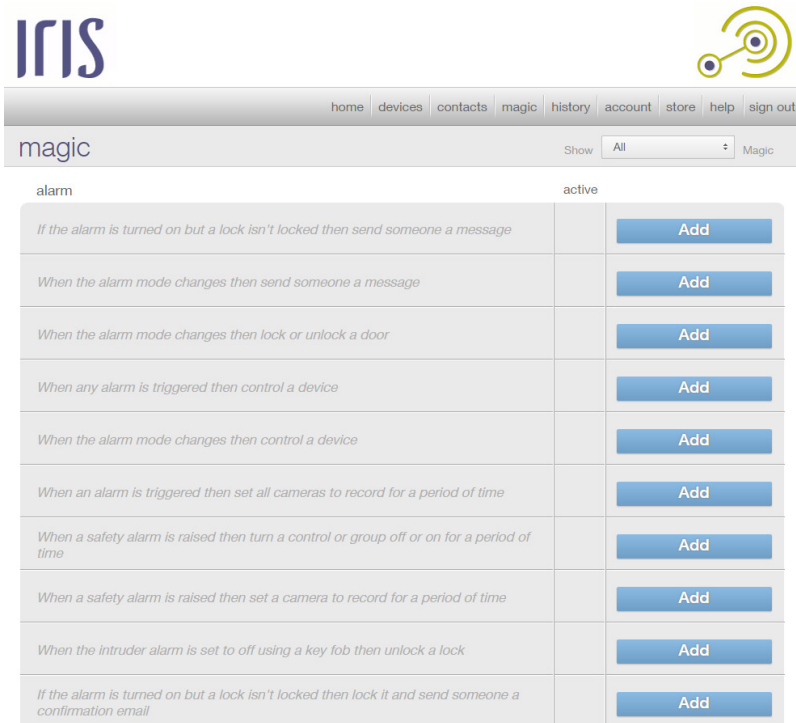
**FIGURE 15-13**

WigWag connected home system (image: WigWag)

Triggers can be directly measured with sensors (e.g., the front door is open), inferred from sensors (the user's smartphone is not at home, therefore the user isn't), or require some combination of sensor data and interpretation (electricity usage is currently 4kW and this is unusually high for this time of day). Some actions might require more than one trigger or action (e.g., *if* the liquor cupboard is opened *and* neither Mom nor Dad is at home, *then* text both of them). Or an action might be conditional on an ongoing state instead of a one-off trigger (e.g., *while* electricity is less than [x] cents per kWh, run the tumble dryer).

Smart home systems like Comcast XFINITY Home and Lowes Iris have long offered similar interfaces for creating home automation “rules” and “modes,” although not necessarily allowing devices to link in with third-party services.

Rules are typically trigger-action commands—for example, “when I turn the bathroom light off, then turn off the speakers in the bathroom” (see Figure 15-14).



**FIGURE 15-14**  
Rules in Lowes Iris

Modes are generally just grouped actions or macros required in a particular context. For example, activating holiday/vacation mode will turn down the heating, set the security alarm, and turn on a lighting pattern meant to simulate occupancy (see Figure 15-15). It's important to differentiate between modes, which are ongoing states, and macros, which are shortcuts that execute multiple commands.<sup>10</sup> Vacation mode may be an ongoing state, but “at home” mode (used when you get home) is logically more of a macro. Some system designers mix the two. But attempting to make macros modal can get very messy.

<sup>10</sup> The concepts are different, but the nomenclature isn't standardized. So you might hear macro-like functions described as modes. As if things weren't confusing enough already.



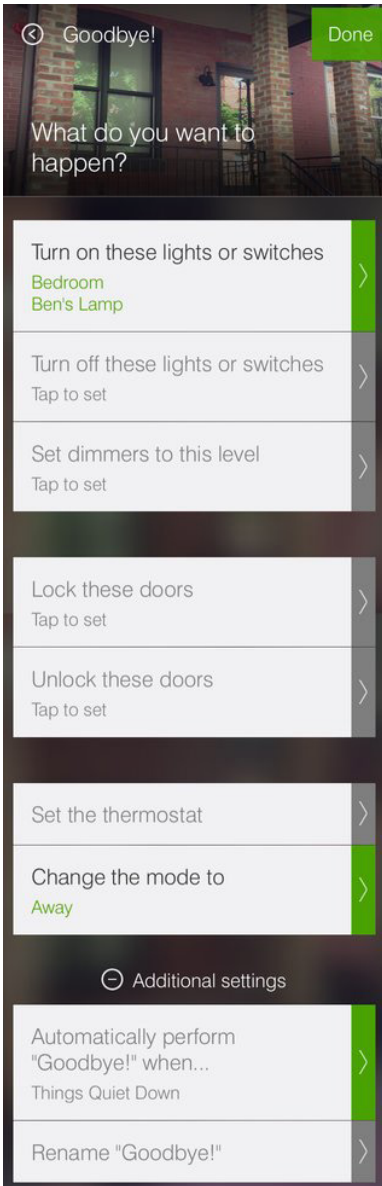
**FIGURE 15-15**

Applying rules to modes in Comcast XFINITY Home (image: Comcast)

Imagine you turn on “movie” mode, which dims lights and turns on the home entertainment system. Half way through, your housemate wants to grab a drink and go to the bathroom so turns some of these lights back on. Is movie mode still “on” despite some of the conditions of movie mode no longer being true? Are you in a modified form of movie mode with some functions overridden? How is this shown on the UI? You can easily end up in a mess of states and overrides.

It’s far easier to treat these things as macros. Macros turn a bunch of devices to some predetermined settings. If you want to change the state of those devices, you can. If you want to rerun the macro, you can. Modes set the system to an ongoing state. You then have to deal with the logical problems of changes being made to that state.

SmartThings treats modes and macros separately. The “Hello, Home” feature allows users to create macros of combined commands, such as “Good morning” (waking up) and “Goodbye” (leaving home). Macros can be used to activate modes (the defaults are “Home,” “Away,” and “Night”). So when the “Goodbye” macro runs, the home is set to “Away” mode (see Figure 15-16). Depending on the current active mode, a device may behave differently, for example, a motion sensor might turn on a light in “Night” mode but activate an alarm in “Away” mode. This is more logical than making everything a mode. But as the number of devices in a system mount up, and users configure their own macros and modes, it can become very complicated to keep track of all the interrelationships between devices, macros, and modes.



**FIGURE 15-16**

The Goodbye macro in SmartThings, which triggers “Away” mode (image: SmartThings)

Our personal opinion, based on small-scale consumer testing, is that grouping functions as macros makes a lot of sense. But the effort of configuring modes, and the potential confusion they can cause, is

currently far greater than the perceived benefit for most people, with the possible exception of holiday/vacation mode. Romantic mode (dim lights, raise thermostat temperature, turn on soft music) is common but perhaps the most jarring. Unless you and the person you wish to impress are both die-hard smart home enthusiasts, the effort involved in setting up your romantic mode may make you seem as if you have a bit too much time on your hands. Or worse, you are a bit of a lothario who requires automation because of the high frequency of your romantic encounters. Neither gives the impression of spontaneity or specialness that many people consider to be elements of romantic experience.

Just to make matters more complicated in the connected home, lighting systems often have another way of grouping controls: scenes (see Figure 15-17). These are akin to macros and modes. A home system of interoperating subsystems and devices could easily have all three of modes, macros, and scenes, possibly implemented slightly differently on each subsystem. This could easily become overwhelmingly complex.

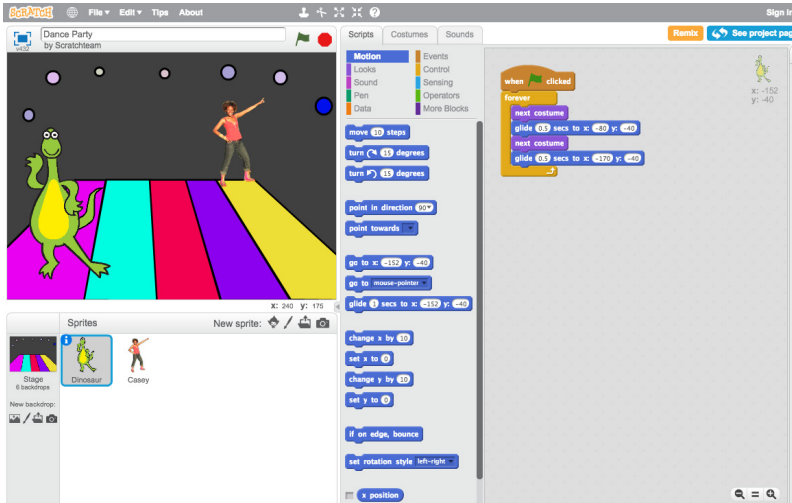
At some point, it's likely that these models will begin to standardize. But it's far from clear what the eventual standard will be.



**FIGURE 15-17**

Control4's MyHome Android tablet app, showing lighting scenes for a kitchen (image: Control4)

Academic researchers have explored various approaches over the years to make programming easier and more concrete for end users. The Scratchable Devices project from Rutgers University used a UI based on MIT's children's visual programming tool Scratch (see Figure 15-18) for configuring many home devices.<sup>11</sup> Other projects have used visual metaphors such as magnetic poetry,<sup>12</sup> jigsaw puzzles,<sup>13</sup> or even physical blocks.<sup>14</sup>



**FIGURE 15-18**

A game in Scratch<sup>15</sup> (image: MIT)

Sometimes the focus of these is device-centric; occasionally as in the magnetic poetry example, attempts are made to allow users to structure commands around people, tasks, or goals.

11 <http://scratchabledevices.com>

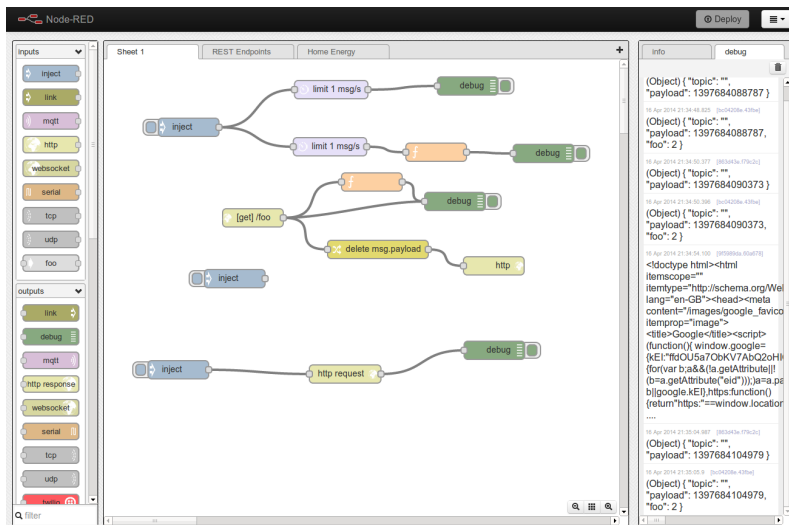
12 Khai N. Truong, Elaine M. Huang, Gregory D. Abowd, "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home," *Proceedings of UbiComp 2004*, <http://www.elainehuang.com/ubicomp2004-camp.pdf>.

13 Yngve Dahl and Reidar-Martin Svendsen, "End-User Composition Interfaces for Smart Environments: A Preliminary Study of Usability Factors," in *Design, User Experience, and Usability: Theory, Methods, Tools and Practice*, ed. Aaron Marcus (New York: Springer, 2011).

14 A.F. Blackwell, (2002). "What is Programming?" *Proceedings of PPIG 2002*, 204–218. <http://www.ppig.org/papers/14th-blackwell.pdf>.

15 <https://scratch.mit.edu>

NodeRED<sup>16</sup> from IBM is worth a look as an example of visual programming, although it is designed to widen the audience for programming rather than to create a mass-market end user tool. This is a free, open source visual coding tool specifically designed for the Internet of Things, based on Node.js (a server-side JavaScript platform). It enables users to stitch together web services and hardware by replacing common low-level coding tasks (such as getting a service talking to a serial port) with a visual drag-and-drop interface. Components are connected together to create a flow, and much of the code needed to link them together is generated automatically (see Figure 15-19).



**FIGURE 15-19**  
NodeRED (image: IBM)

An alternative approach to visual or written syntax is programming by example: demonstrating to the system what you want it to do. Microsoft employed this approach with HomeOS.<sup>17</sup> Users can record macros to activate sequences of actions, such as turning on lights when a door is opened. This allows users to set up simple macros without the need to learn an abstract visual or verbal representation of the system.

16 <http://nodered.org/>

17 <http://research.microsoft.com/en-us/projects/homeos/>



It sounds simple, but it's not without issues. What if you put the house in learn mode, and are trying to demonstrate a sequence of actions you want the house to remember when someone else in the house does something unexpected and the system learns that too? Walking into the living room now doesn't just turn on the lights, but also opens the cat flap and puts *SpongeBob SquarePants* on TV.

And programming by example does not get around the need accurately to predict future needs, nor allow users to set up multiple triggers, nor specify conditions under which a macro should or should not be triggered.

It's also not clear what happens when the user wants to edit a macro, or just find out which macros exist: do they still then need to engage with an abstract representation?

This highlights a trade-off in end user programming: making the tool easier to use generally limits its *expressibility*, or flexibility to support a wide range of commands and logic. Full-featured programming languages may be highly expressible and able to do a wide range of things, but have a steep learning curve for novices. For each tool, you need to define an appropriate balance, depending on what the user may need the system to do, and their prior knowledge and ability to configure it.

As discussed previously, the complexity in programming-like activities is not just in understanding how to instruct the system to do what you want. It's also in diagnosing and fixing bugs, and ensuring there are no unintended consequences of your commands.

The Whyline programming tool (a research project from Carnegie Mellon University) allows users to ask “why did” and “why didn't” questions about their program's output.<sup>18</sup> For the purposes of research, the project focused on users controlling a Pac-Man character around a virtual world. But an approach like this could be hugely valuable in, for example, a home automation system. Users would be able to get answers to questions such as:

- “Why didn't the dishwasher run this morning?” (perhaps the smart electricity grid turned off the supply to that nonessential device at a time of peak demand)

---

<sup>18</sup> <http://www.cs.cmu.edu/~NatProg/whyline.html>

- “Why is the hall light flashing on and off?” (perhaps it is receiving conflicting instructions from security and energy saving apps)

For predicting future actions or testing, simulation is a possible approach—for example, the ability to test your new smart home rule in a sandbox environment and identify any clashes with other rules safely before unleashing them on the main system.

**Is programming the right model?** We’ve proposed that configuring a system to do something in another place, or at another time, is a form of programming, and that the Internet of Things is largely concerned with these things. That means that focusing on designing programming-like experiences is a key challenge for IoT. As programming is mentally complex, ways need to be found to make it easier for a wider audience.

But the success (or otherwise) of a programming-like experience for a particular problem isn’t just about whether users are able to understand their tools (which we tend to assume is a function of some combination of innate ability, learned skill, and the design of the tool). It’s also about attention.

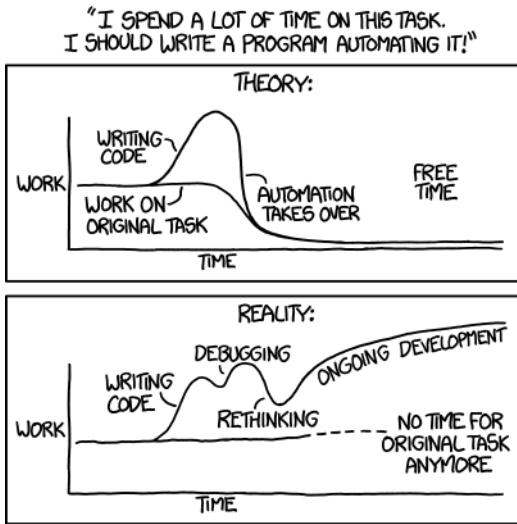
In a world that is mostly powered by computers, it’s important that we empower as many people as possible with the logical and practical skills needed to understand and control those computers. Not everyone need be a professional programmer, but all of us benefit from a better grasp of computational thinking. Introductory tools like Scratch can be very helpful here. But however easy the tools or good our training, programming-like activities that are supposed to make life easier through automation can paradoxically make intensive demands on our attention to set up and manage them.

As discussed in Chapter 4, this effort needs to be justified in terms of the value to consumers, who tend to favor focused products over tools. But we can’t preconfigure products to know exactly how to work in complex multidevice systems. Some configuration is inevitable; it will be conceptually like programming, and it will need our attention.

For many of us, attention is in short supply. Most consumers probably have at most a couple of burning issues in their lives in which they are sufficiently interested or invested to expend time in advanced configuration or system administration. Alan Blackwell frames the user’s

dilemma as the “investment of attention”: should he spend time figuring out how to program a system, considering how hard it will be, how much time it will take, and what the benefit or risk might be? Or is it more efficient or less risky just to do it manually (see Figure 15-20)?

By embedding programming-like activities in daily tasks, we risk disenfranchising a percentage of the population who lack the skills or time to cope with them.



**FIGURE 15-20**

Automation, by XKCD<sup>19</sup> (CC license)

At the times when this may be too limiting, there is another approach: pay someone else to do it. Consider household plumbing: some people do their own, some people enlist the help of a friend or relative with good DIY skills. Anyone with a job beyond their own (or their friends’) abilities hires a pro.

Translating this to IoT, users might configure their own behaviors or buy apps for their homes or other consumer needs, or find shared “recipes” on services like If This Then That. But these would still require a considerable degree of customization to fit each user’s personal circumstances. For the most daunting problems, they might pay someone

<sup>19</sup> <http://xkcd.com/1319/>

else to understand their needs and set it up the way it should work, as is the case with the Time It Right Orthodox Jewish home automation example briefly described in Chapter 4. We may see the emergence of new categories of home configuration and support technicians.

## GIVING MACHINES CONTROL

In this section, we'll look at the opportunities and challenges of putting the system in control.

### Make the system smarter

Some researchers have argued that making programming more usable is not feasible, and instead, we should make smarter systems that learn and adapt to the user's needs. Popular fiction loves to characterize AIs as anthropomorphized servants, like *The Jetsons'* robot maid Rosie<sup>20</sup> or J.A.R.V.I.S. in *Iron Man*.<sup>21</sup> But most examples of intelligent or adaptive systems right now are more prosaic, specialized devices like thermostats and vacuum cleaners (see Figure 15-21).



**FIGURE 15-21**

The iRobot Roomba 870 vacuum cleaner (image: iRobot)

Making devices smarter offers far greater potential than remote controls and programming. Nicholas Negroponte, the founder of MIT Media Lab, recently argued that the kind of remote control-based IoT products coming out right now are neither innovative nor intelligent:

I look today at some of the work being done about the Internet of Things, and I think it's kind of tragically pathetic, because what has happened is people take the oven panel and put it on your cell

---

<sup>20</sup> <http://thejetsons.wikia.com/wiki/Rosie>

<sup>21</sup> <http://marvel-movies.wikia.com/wiki/J.A.R.V.I.S.>

phone, or the door key onto your cell phone, just taking it and bringing it to you, and in fact that's actually what you don't want. You want to put a chicken in the oven, and the oven says, "Aha, it's a chicken," and it cooks the chicken. "Oh, it's cooking the chicken for Nicholas, and he likes it this way and that way." So the intelligence, instead of being in the device, we have started today to move it back onto the cell phone or closer to the user, not a particularly enlightened view of the Internet of Things.<sup>22</sup>

### **What intelligent systems can do**

A simpler autonomous system may have a goal, be able to observe specific data inputs, recognize patterns in the data, and execute a limited range of functions in response.

For example, a thermostat might have the goal of maintaining a comfortable temperature in a home when it is occupied. It would monitor sensor inputs, such as motion, temperature, and lighting, or the location of mobile phones, and activate behaviors in response, such as turning the heating or AC down when there is no one at home. It might also, over time, learn the patterns of occupancy of the home and adjust its own schedule accordingly (as the Nest thermostat does).

A robot vacuum cleaner, like the Roomba, has the goal of cleaning the entire floor of a room. By traveling around the room bumping into things, it learns the layout of the room and its obstacles over time. This allows it to calculate the optimum path for cleaning the room next time around, and know when it has finished.

A more advanced artificially intelligent system might have some broader knowledge about the domain in which it operates, and the ability to learn about the user's needs and intentions. This is the stuff of the kind of smart home dreams that have been around since Mark Weiser, and before. For example, if a "smart" home system noticed that a user had an early meeting on the other side of town and that rainy weather was likely to result in more traffic congestion, it might decide to adjust her alarm clock to wake her up earlier and turn the hot water on for an earlier shower, factoring in the amount of time it has observed she typically takes to get ready.

---

22 Nicholas Negroponte, "A 30-Year History of the Future," TED 2014, [https://www.ted.com/talks/nicholas\\_negroponte\\_a\\_30\\_year\\_history\\_of\\_the\\_future](https://www.ted.com/talks/nicholas_negroponte_a_30_year_history_of_the_future).

There's definitely scope for autonomous systems to help us out with some of the tasks that humans are simply not very good at or find boring (such as controlling heating, and vacuuming). Research projects such as Intel's Proact<sup>23</sup> and MIT's house\_n<sup>24</sup> have also explored the use of intelligent systems to monitor the day to day of people with dementia, through tagging household items with sensors, learning their "normal" patterns of use, and alerting users or caregivers to anomalies and risks, such as ovens left on.

The fundamental challenge is that it is very hard for computers to understand humans, and behave in "common sense" ways. For an autonomous system to understand how to act in any situation, it needs to know how to infer and use context appropriately. Humans are able to understand context holistically. We take in a huge amount of information about a given situation (much of it subconsciously) and intuit what is important and how best to act. Getting a computer to do the same thing requires us to tell it, in very concrete and logical terms, what to look for, how to interpret it, and how to act.

This is extremely hard to get precisely right, and even a very smart system will sometimes get it wrong. When it does, it will be intensely annoying at best and disastrous at worst. Cold showers, unpalatable meals, or unnecessarily early awakenings while the system learns your preferences are one thing. Being locked out of your own house because your home thinks you are an intruder or being unable to make your car take you to hospital because it thinks it's a work day would be utterly unacceptable. But getting it right can take a level of intelligence that is prohibitively time consuming and expensive to develop, just to do something that is common sense to a human.

Scott Jenson uses the example of sensor-based lighting in a bedroom to illustrate how complex inferring the right context of a simple task can be. It sounds simple to turn on the lights when you walk into a room, and most of the time that might be useful, but the exceptions to the rule are very hard to get right:

---

23 [https://homes.cs.washington.edu/~matthai/pubs/pervasive\\_sharp\\_o4.pdf](https://homes.cs.washington.edu/~matthai/pubs/pervasive_sharp_o4.pdf)

24 [http://web.mit.edu/cron/group/house\\_n/](http://web.mit.edu/cron/group/house_n/)

Problem: I walk into the room and my wife is sleeping, turning on the lights wakes her up.

Solution: More sensors: detect someone on the bed.

Problem: I walk into the room and my dog is sleeping on the bed, my room lights don't turn on

Solution: Better sensors: detect human vs pets

Problem: I walk into the room, my wife is watching TV on the bed. She wants me to hand her a book but, as the room is dark, I can't see it.

Solution: Read my mind.<sup>25</sup>

Humans are extremely good at this type of context inference, and will not tolerate “silly” errors from machines, especially not in their own homes where we expect things to be adapted to our needs. Intelligent systems may have to be so smart to work effectively that the effort of building and teaching them will only be justified if they provide significant value. But in a consumer setting, some of the more advanced ideas are overkill for the amount of value we get from them. Most of us don't need or want hard AI just to turn on a coffee machine. And maybe the advanced AI would resent doing something this menial:

Here I am, brain the size of a planet, and they ask me to take you to the bridge. Call that job satisfaction? 'Cause I don't.

**MARVIN THE PARANOID ANDROID,  
THE HITCHHIKERS GUIDE TO THE GALAXY<sup>26</sup>**

And the learning period the system requires to have any hope of getting it right can require additional user effort and vigilance to teach the system “normal” behavior. For example, Nest thermostat users must control the system manually for the first week or so. If a heating system is using electrical activity above baseload as a proxy for presence or activity, then home occupants will have to avoid running the dishwasher or tumble dryer when they are out or in bed during the learning period, or it will think they are in, up and about.

---

<sup>25</sup> Scott Jenson, “Home Automation is an EasyHard Problem,” February 2014, <http://jenson.org/easyhard/>.

<sup>26</sup> Douglas Adams, *The Hitchhiker's Guide to the Galaxy*, (London: Pan Books, 2009 [1979]).

However, we can expect the accuracy of machine learning to improve in the future. Huge investments are being made into research in autonomous technologies, such as self-driving cars, as companies like Google see this as a valuable market opportunity.

A possible middle ground that could help alleviate some of the pain of configuring complex systems is a mix of user control and autonomy. An intelligent heating system might monitor occupancy and set its own schedule, but also give users control to turn the heat up or down, or demand more hot water, and over time the system would adjust better to their needs. It might also take into account the fallible human psychology in place around heating: for example, the tendency to crank the thermostat up high when feeling cold in the false belief the house will warm up faster. The system might acknowledge the user's action and give the impression that the heating had been turned up a lot, but actually only turn it up a tiny bit. Or if the house was already in the process of warming up to a schedule setting, it might quietly ignore their command. Autonomous systems can be useful in bridging the gap between what users actually need and their ability to configure the tools at hand to achieve that, without taking away control.

A system might also observe the user taking an action repeatedly (e.g., always turning off certain appliances at night), and ask whether they would like to automate this in the future. Or a system might notice a pattern of repeated overrides of a smart home rule (e.g., a light that has been turned on as part of a macro is always manually turned off immediately afterward) and offer help modifying the rule.

### **UX risks of autonomous systems**

Autonomous systems can save users time and effort, and do certain types of job better than humans. But of course the user always needs a safe, easy override, in case the system fails or does something strange. And designers need to be aware of some potential user experience pitfalls.<sup>27</sup>

---

<sup>27</sup> For more information, a good resource is: Anthony Jameson and Krzysztof Gajos, "Systems That Adapt to Their Users," in *The Human Computer Interaction Handbook*, Third Edition, ed. Julie Jacko (Boca Raton, FL: CRC Press, 2012).



**Reduced control.** Giving too much control to the system can take control away from the user and prevent them from getting the system to do what they want it to do. The system may not be pulling a HAL and trying to take over, but it will be frustrating and disempowering at best. A smart home might have windows that are opened and closed automatically to manage the internal climate, but if these controls stop working properly they risk opening and closing windows at random. Residents should have an override switch to control the windows themselves, and if all else fails the system should be able to be shut off without losing manual control of the windows.

If the autonomous actions are only occasional, the system could ask permission before acting, for example: “I see you never come home before 5 p.m. but the heating comes on at 4. Would you like me to adjust your schedule?” Or it could notify the user after making a change and give them the option to cancel the action. But if the system is making lots of decisions and frequently asking for permission, this will become intrusive and irritating.

**Reduced predictability and comprehensibility.** For users to be in control, they need a clear understanding of how the system works, what it does, and why it behaves in particular ways in response to their actions or certain situations. It may not be that easy to explain why a complex system controlled by adaptive rules is behaving as it is, especially if it has a limited UI through which to communicate. It may not even be that easy to understand the boundaries of what it can or cannot do.

As the writer Howard Rheingold puts it:

“What child will be able to know that a doorknob that recognizes their face doesn’t also know many other things? We will live in a world where many things won’t work, and nobody will know how to fix them.”<sup>28</sup>

And autonomous systems interacting with other autonomous systems will be especially hard to keep track of.

---

<sup>28</sup> Howard Rheingold, 2002, “Smart Mobs” (also cited here: <http://www.pewinternet.org/2014/05/14/internet-of-things/>).

## Balancing user control with system autonomy

As smart (truly smart, not just connected) devices become more commonplace in the world around us, we will need to figure out how to design systems that allow people to do what they are good at, and machines to do what *they* are good at.

Users should never lose control: there must always be an override. If the override fails, the system should be usable manually. While the system is learning, it should ask the user's permission to act autonomously if that would not overwhelm them with messages (e.g., "I see you tend to wake up at 7 a.m. Would you like me to set an automatic alarm for 7 a.m. every weekday?"). If that creates too much overhead for the user, it should at least be possible for the user to understand everything the system does, why it does it, and to correct any incorrect assumptions in the user model ("Mom does not live here, she just visits occasionally").

Systems could offer to automate routines that are clearly established, like shutting all windows when the last person leaves.

Intelligence can also be used not to change system functionality but to offer adaptive help to the user as they try to complete unfamiliar tasks, or to adapt the interface or information presented to better fit their needs at that time or abilities.

We should also not forget the emotional dimension to automation. The fear of the machines taking over is almost as big a barrier as the actuality. Users often overestimate how much a system actually knows about them and how much control it really has. Even if a system isn't really that smart or capable, it must be made to feel nonthreatening. Part of that is about being transparent about the information being captured, and what is being done with it, as described earlier. Another part is about the emotional design of the system: the form and appearance of the devices, and the conversation it has with the user.

In robotics and 3D animation, the term "uncanny valley" is used to describe the unsettling effect of faces that are almost, but not quite, human. A face that is clearly *not* human feels far less alien and threatening. The designer Matt Jones draws the parallel with autonomous systems, suggesting that in trying (and failing) to be smart like a human, they will fail and feel creepy. Instead, he proposes designing

things that we know to be less smart and capable than ourselves, that help us retain our sense of control and not feel threatened. He calls this principle “be as smart as a puppy”:

Making smart things that don't try to be too smart and fail, and indeed, by design, make endearing failures in their attempts to learn and improve. Like puppies.

**MATT JONES**<sup>29</sup>

## Approaches to Managing Complexity

In this section, we'll consider some emerging methods by which complex systems of many interconnected devices can be made more manageable to users.

### HOW UX IN THE PLATFORM SCALES UP

In Chapter 12, we talked about designing platforms: software systems that support the development of multiple applications. In that chapter, our focus was on the initial design challenges of a growing platform: common design components to make the UX and UI design feel coherent and consistent.

As a platform scales to support more and more devices and more applications (or services), you'll find it's no longer practical to hardcode in all the interrelationships between those devices, applications, and users. You need some way to automate the work of coordinating all these elements. And you need flexibility in the system to respond to devices, functions, and combinations of devices and functions that you haven't anticipated.

This will take the form of data models and logic that specify how devices and software services should work together. In architectural terms, this will sit in the platform “middleware.” Your system engineers will have many issues to address in the platform logic, not all of which will have a direct impact on UX. But for a complex system, your ability to create a compelling and useful design is based on building blocks that the platform creates for you. The right platform logic will enable you to create

---

<sup>29</sup> <http://berglondon.com/blog/2010/09/04/b-a-s-a-a-p/>

a UX architecture that fits user needs well. If the platform logic doesn't help you put the pieces together in the right way, it will be difficult or impossible to mitigate this with UI design.

Earlier we talked about some system constructs for a connected home. There might be services (the smoke alarm, lighting, energy usage), devices (motion sensors, cabinet sensors, window shade controls), controls (on/off, up/down, timers, set/unset), various forms of notification (alarm, message, status), users or contacts (with presence status or location information and varying system permissions). There might even be multiple properties or gateways. We also presented the variety of ways user needs might be thought of: around devices, people, locations in the home, times or states (away, night), optimization (comfort, efficiency), controlling access to dangerous or private things.

Some of the specific use cases that you might need to support could include:

- The user needs a way to keep track of everything that happens on the system and why it happens (e.g., what rules are running, and what assumptions is the system making when it acts autonomously?). For example, “At 8 p.m., the random security lights app was activated, because it was dark outside and no one was at home.”
- When the user adds a new device, the system suggests sensible ways of incorporating it into the system, and sensible ways it can work with other devices. It already knows that lights can be useful for security, and that there is no point offering to turn on the washing machine when the burglar alarm is activated.
- The user can address devices and functions in ways that make sense, for example, turning off downstairs lights or entertainment devices or locating Jessica's toys or monitoring the baby's room or Granny's apartment.
- The user needs a way to see and edit who is allowed to do what on the system. The builder needs to be able to come and go for a week, but shouldn't be able to pass on their access to anyone else.
- The user needs a way to look up a device, and find out which applications, services, or rules are controlling it, and which other devices it is coordinating with. Why is that light flashing on and off at 8 p.m. every night?

- The user wants to add a rule to turn off nonessential devices at night to save energy. Will this break anything, conflict with any other rules, or otherwise cause unexpected consequences?

There is no one way of organizing system functionality in a traditional hierarchical UX that supports all of these needs. You can force people to think in terms of a simplified schema (i.e., organizing the house by rooms), but it will feel pretty clunky and unnatural once you get beyond a certain number of devices. You'd prefer to design something that's flexible enough to support the way they think, but just giving them every possible piece of functionality would risk overloading them with options. So, you need some kind of organizational logic to underpin the UX and bridge the system makeup with user needs. This means some mix of system intelligence and data models to describe the devices, data, and system functionality, and how those things can be combined in different ways to meet different needs. Think of this as UX design for the platform layer.

The goal of your platform UX is to create a system that not only feels consistent and coherent, but also behaves in common sense ways. But computers don't have the knowledge and experience of the world that we do or know how to reason with it, unless we program it in. If a user has to do all that themselves, they might well decide that the system is too much hassle to use. So we, as system designers, have to help them.

The UX design work that goes into a platform covers two key areas:

- Provide common design elements that help create consistency and coherence in the UX across different functions and applications (as we saw in Chapter 12).
- Data models and logic about how all the components in the system relate to help it behave smarter. These can be used to organize controls and interpret data in smarter ways. If you want to be really flexible and provide lots of different views, you'll need some metadata to model the relationship between the devices, data, context, and people. Who is "Jimmy"? What is in the living room? Which devices and states relate to watching a film?
- An autonomous system might also have broader knowledge about user needs and the context in which it is operating.

If you're designing services that run on top of an existing platform, you'll need to understand what it does and what standard ways of handling things exist. If you're designing things that run on a new platform, you may find yourself having to think about general platform components and logic as well as the specifics of your new application. This can be hard: you have to think about both the big picture and the details. You have to create generalizable components and logic that work for many use cases and situations. But you also have to design for specific use cases/situations or the design won't be good for those things.

## DATA MODELS

Data models describe connected things to applications, driving discovery and linking.

**IOT RESEARCHER MICHAEL KOSTER<sup>30</sup>**

In this chapter, we use the term “data model” as a catch-all for a range of concepts that might be used in system software. These might describe system components, including devices, users, and services (in the sense of applications that may run on your platform). They may describe the data collected by your system, and they may define the relationships between different system components. Or they might provide wider domain knowledge about user needs or the context in which the system operates.

Data models allow us to configure the system to behave in smarter ways. In a small system, you could code in every single possible interrelationship between the different people and devices and services. But as we saw earlier (Figure 15-3) that would soon become an unmanageable headache. Instead, you create an abstract representation in software that automates some of the work of orchestrating those components.

Data models can be used to improve interoperability, organize controls, and interpret data in smarter ways, better suited to bigger systems. An autonomous system might use domain knowledge about user needs, activities, and context to determine how to act.

---

<sup>30</sup> <http://iot-datamodels.blogspot.co.uk/>

Data models can make or break a user experience. They give the system a “mental model” of how things work together. If there is no abstraction in your system, you’ll find that as it scales and more devices are added, you’ll feel increasingly constrained in the functionality you can offer. Abstractions allow you to make the system more general purpose. Instead of viewing a camera as just part of a security alarm, you can treat it as a way to capture media, freeing it up for alternative uses such as monitoring a baby, or a pet’s feeding habits. And the right data models will support you in designing a system that is a good fit for the user’s mental model of how things *should* work together, even as it scales.

If IoT is to scale, some kind of data models are essential. The question is not whether data models are needed, but which ones. Commercial platforms (such as Thingworx<sup>31</sup>) may have their own proprietary data models. Some people are working on shared standards to drive interoperability, but this is still a nascent technical field.

Next, we’ll look at some ways that data models might be used to support the design of IoT systems. These may include:

- System-generated metadata about devices. This might include information about the device’s type and capabilities, and the data it gathers. Having standard ways of sharing this information enables devices to describe themselves and their data, so applications (and users) can find useful and relevant devices and know how to work with them.
- Metadata about the device context, which may be generated or edited by the user. This might include its name, location, who owns it, or a description of the purpose for which it is used. Applications can use metadata like this to present devices and functions in smarter ways to the user, or better interpret the data they are measuring.
- A domain model that describes your system’s logical components, and how they should work together to support particular user activities or in particular contexts.

---

<sup>31</sup> <http://www.thingworx.com/>

- Wider domain knowledge about the activities, or the context of use. An intelligent system might use this to learn how to behave in different situations.
- Knowledge about the user, to help the system adapt.

Bear in mind that these aren't methods that are, as of yet, commonly found in mature products. But they are an indication of the kind of semantic technologies that will be needed to make complex systems coordinate in ways that make sense to humans.

### **System-generated metadata about devices**

Right now, much of the effort around data models is focused on developing ways to describe devices to applications. This enables devices to describe what kind of device they are, what physical properties they sense or what actuation capabilities they may have, and who made them.

A proprietary platform may make use of device models like this, but here we will focus on open examples aimed to improve interoperability. For example, the IPSO Alliance (an open organization that promotes the use of IP networking for smart objects) has published an object model that aims to support interoperability between connected devices and software applications on other devices and services. The first version of the model describes 18 standard, common smart object types likely to be used by a variety of applications. Figures 15-22 and 15-23 show the data that can be captured about two object types: a temperature sensor and a lighting control.

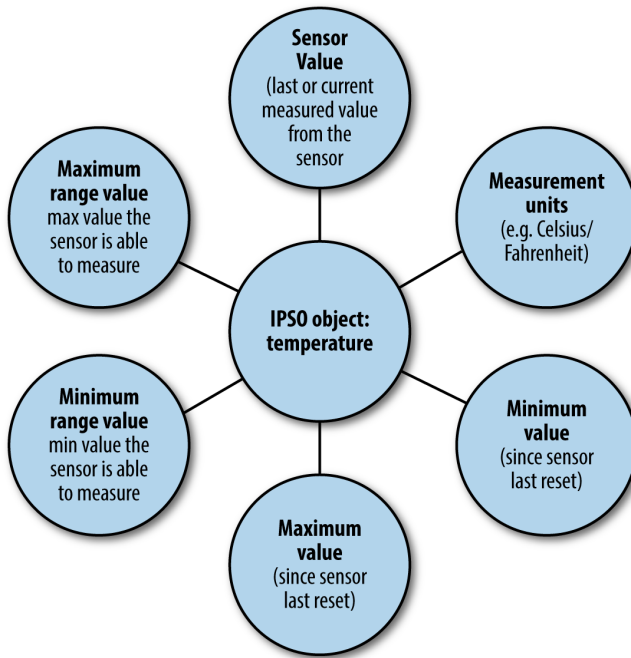
The W3C Semantic Sensor Network Ontology is a similar initiative in this area. This is a common framework for describing the sensing capabilities and data measurements of sensors.<sup>32</sup>

This kind of metadata helps aids interoperability. When you add a device to a network, applications (and potentially other device) on the network are able to understand what it is, what it can do, and how to talk to it.

---

<sup>32</sup> <http://www.w3.org/2005/Incubator/ssn/>





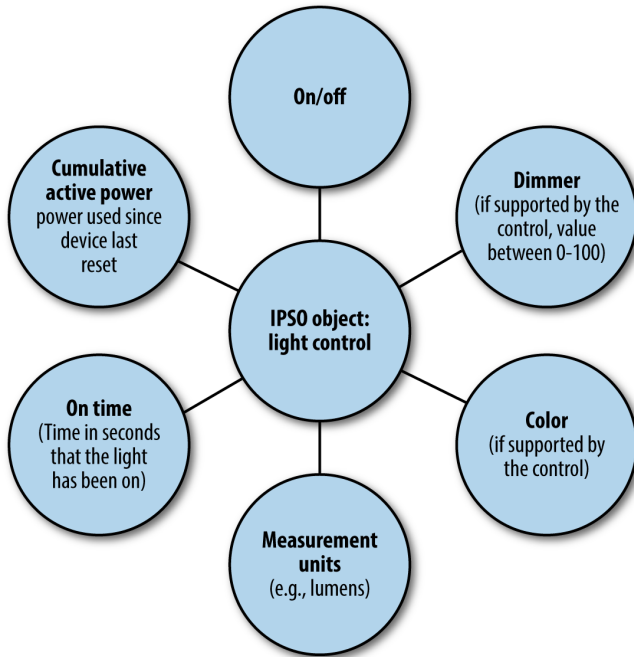
**FIGURE 15-22**

The resources defined in the IPSO object for a temperature sensor (simplified from the full object)<sup>33</sup>

Imagine you add a new light switch in your home. Your system uses a common metadata standard for describing devices. It's as if the light switch is able to say: "Hello, I'm a light switch, I'm new round here. I can be off, on, or x% dimmed and I can monitor the power consumption of the light."

Your system is able to understand what it is saying. It knows what a light switch is, regardless of who manufactured it. It can figure out how the switch can be incorporated onto the system along with any other light switches you already have, added to any applications running on the system that know what to do with light switches and power sensors, and determine what functionality and data to offer the user. It's as if it were responding: "Added new lighting control to lighting application, and power consumption readings to home energy monitoring."

<sup>33</sup> See [www.ipso-alliance.org/](http://www.ipso-alliance.org/) for full version.



**FIGURE 15-23**

The resources defined in the IPSO object for a lighting control (simplified from the full object)<sup>34</sup>

Devices that describe themselves can be used in smarter and more flexible ways. Devices that can describe themselves to applications on the network need not themselves be “smart,” but can work with multiple applications, thus making use of smartness in the network. In the previous example, imagine the home system is also running a security application, which has a lighting feature for making the house look occupied. As you add the device, it might also ask you, the user: “Do you want to add this to your security lighting service too?”

However, other more complex devices will continue to use some intelligence at the edge. A washing machine will continue to need internal timers and controllers; most of us would not feel comfortable with a machine that was entirely controlled by a remote application. So future iterations of object descriptions will need to describe the object’s onboard capabilities and intelligence.

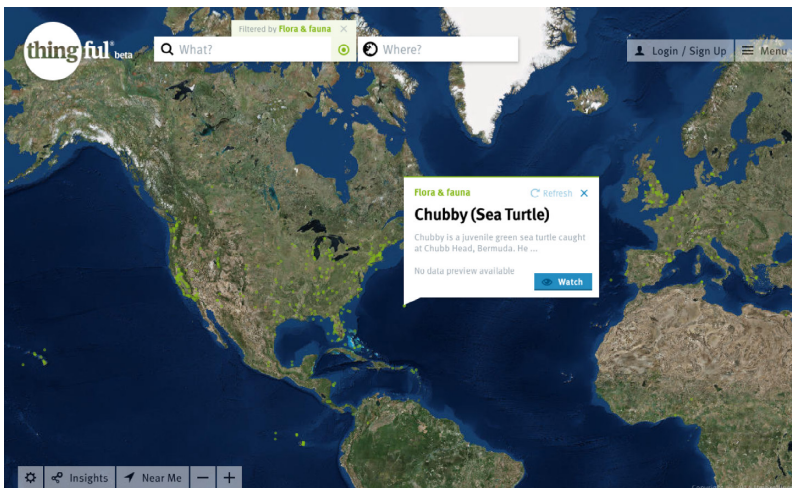
<sup>34</sup> Ibid.

## Metadata about the device context

System-generated metadata allows us to identify a device as being of a particular type and having certain capabilities. But it doesn't tell us anything about a specific *instance* of a device. For example, does the light switch you added to your system live in the living room or hallway, and does it control a wall or ceiling light?

It's often useful to add some descriptive information about particular instances of each device. That might be as simple as a name and where it lives in the house, or it may be important to identify who owns it or for what it is being used. This information might be created or edited by the user, or some data might be added by the system. Systems can use metadata like this to present devices, data, and functions in more meaningful ways to the user.

For example, the Thingful search engine allows users to search for connected devices that share data publicly on the Internet (such as personal weather stations, off-grid energy systems, air quality sensors, and even animal trackers; see Figure 15-24). Crucially, it also allows the owners of these devices to provide contextual information about their device and what they use it for.



**FIGURE 15-24**

Thingful.net tells us that this satellite transmitter belongs to Chubby the green sea turtle, recently seen swimming off the coast of Bermuda

Knowing what a device is being used for allows the system to present it in more meaningful ways. Smart sockets/outlets are an interesting challenge here. What's important to the user is not the outlet strip, it's the device that's plugged into it. The outlet strip may have its own capabilities (on/off and power measurement), but these aren't very meaningful unless you know what is plugged into it. A strip with a timer could be used to turn on a security light, turn off hair straighteners that have been left on for more than 10 minutes, protect a computer from power surges, or control the light in a chicken house used in winter to make hens lay more eggs. If the system knows what is plugged into the strip, it can offer appropriate functionality, but the user would be responsible for keeping this data up to date if devices were unplugged or moved.

Adding metadata about specific devices also allows us to organize them. Some types of system may offer a fixed schema of predetermined metadata types for doing this. For example, ZWave devices can be assigned to rooms and scenes. A scene is a group of different devices that can be controlled with a single, macro-like command, for example a "TV scene" might dim the lights to 50% and turn on the home entertainment system.

But fixed schemas can lack the flexibility to meet all user needs. For example, not all user tasks in the home fit the room or scene device-grouping model. As we've previously seen, some may be oriented around activities (e.g., cooking dinner, going to bed), people, times, more complex ways of describing location (e.g., the west side of the house), or random and very personal categories such as things that are on loan from someone else, things you turn off at night, or things visiting children shouldn't touch.

Allowing users to define their own schema (e.g., by tagging) would provide more flexibility, allowing users to address the group of things they want easily at any point, carving groups up in different ways. But user-defined schemas are prone to inconsistencies and inaccuracies. A mix of some standardized metadata (room, device type, device name) with a user-defined schema is one way to go. Users can use as much or as little of it as they see fit.

Categorizing devices can be useful, but runs the risk of adding admin overhead to the system. If users see it as too much effort they won't bother to use it.

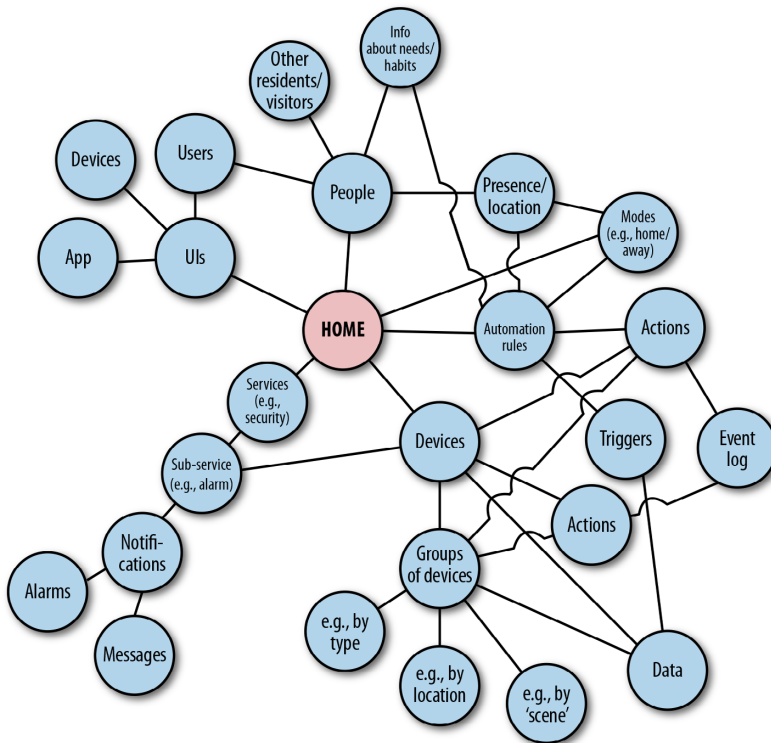
## Data about the system

Users want systems to do reasonable things but the systems have no basis for reason.

**LOUISA HEINRICH, STRATEGIST AND INSTIGATOR<sup>35</sup>**

A system domain model maps out the logical concepts and components in the system, and how they interrelate. It isn't a model of the real world, but it helps map system functionality to the real world.

A high-level domain model of the system might define the conceptual components and interrelationships needed in the system to knit together devices, software services, data, functions, and ways of organizing those. (See Figure 15-25.)



**FIGURE 15-25**

A simplified model of key concepts for a connected home application showing devices, services, grouped actions, and users

<sup>35</sup> In conversation, <http://www.louisaheinrich.com/>.

A more specific domain model might describe the concepts needed within a particular application area. For example, a security system might have concepts such as:

- Users (permanent approved people, temporarily approved people, and unknowns)
- Identifiers (ways of identifying users, such as fobs, keypad codes, or mobile phones)
- Sensors (motion sensors, door/window sensors)
- Actuators (alarm sounder, cameras, door locks)
- Home states (home/away/vacation) and alarm states (set/unset/triggered)

Domain models can exist on multiple levels, in much the way that a web browser provides a framework but can be extended through plug-ins to support specialized tasks. There might be a general smart home domain model that knows about light and energy, to which you might choose to add a maintenance plug-in for your boiler, which does detailed diagnostics and integrates with a plug-in that monitors the efficiency of your heating system.

### **Data about the wider context**

Domain models can represent knowledge about the world, rather than the system itself.

A knowledge model might provide a way to convert data from different devices or systems that model it in different ways. In Chapter 10, we described the example of fitness tracking devices that measure activity in different ways: minutes of activity, number of steps, or in Nike's case, "Fuel." Or light sensors may measure light in candela or lumens. A knowledge model (or ontology) is a way of giving the system knowledge about light, or activity tracking. It can then understand that certain types of data, although they appear different, are actually measuring the same thing, know how to convert between the two, and use the data in meaningful ways.

Knowledge about the activities the system supports can be used to inform the reasoning of intelligent systems. For example, an intelligent alarm system might use knowledge of the different people in the house, their relationships and capabilities, likely location in the house

at certain times, and the differences between genuine burglary and accidentally tripped sensors. If it “knows” that there is a toddler in the second floor bedroom who regularly climbs out of their cot and wanders around at night it might be able to respond to a bedroom door being opened and motion on the landing by sounding a gentle alert in the parents’ room. If it detects a ground floor window being forced from the outside, it might raise a full alarm.

### **Data about the user**

Another form of data about the wider context is user modeling: the system building up an understanding of the user. This could be a learning system, in which the system tries to learn about the user’s behavior and intentions over time to respond appropriately. Intelligent heating controllers try to learn when the house is occupied and what temperature the occupants find comfortable. Elderly care systems could potentially monitor behavioral patterns and identify anomalies to spot when someone is ill or a dementia sufferer is behaving erratically. A system might even try to understand what the user knows, to tailor suggestions and system guidance better.

User data could also be a persistent and portable set of information about you as a user that you control, used to mediate your interactions with things, people, and multiple systems. That might include your schedule, contact list, and preferences (e.g., enlarging fonts for aging eyes, or modifying displays to suit red/green color blindness).

### **How much modeling do you need?**

When you’re designing data models, you need to strike a balance between using very powerful but time-intensive formal methods, like ontologies, and lighter methods that do only as much as you need. The cost and effort has to justify the value. Most of the web gets by without semantic web technology, because in many cases the value and findability of the content is acceptable enough as is and the effort of describing web content semantically is very high.

But IoT affects the way the real world behaves, not just how content on screens is linked. The potential impact of dumb devices that don’t know how to coordinate with each other in sensible ways is far greater. If we don’t stitch connected devices together properly, there’s a real danger that the real world will become more difficult to use and more obstructive to our daily tasks than helpful. Semantic technologies for

IoT, where practical, are not just about describing things but helping real-world systems coordinate in sensible ways. The sheer increase in data and complexity of IoT may make it necessary to imbue systems with some knowledge about the world.

## Summary

Adding just a few extra devices can add vastly more potential interconnections. Making the system do meaningful and valuable things for users means forming the right connections, and helping devices and applications coordinate in the right ways.

Just keeping track of all the devices is a big challenge. UX designs based around device controls won't scale to bigger systems. The UX will need to be organized around the user's broader needs and activities, not just device controls.

IoT breaks direct manipulation: the prevailing trend in UX and UI design of the past 30 years. The consequences of the user's actions are often displaced in space or time, and sometimes end up having unexpected effects when devices are repurposed for different uses. This means that using IoT systems will often be conceptually more like programming. This requires users to anticipate their own needs and configure the system to meet them, but also to deal with unforeseen future conditions that may make the system behave in unexpected ways. Designing systems to make this process usable to the widest possible audience is a challenge.

Handing control to autonomous systems is one way to manage the complexity. But it is very hard to get computers to anticipate our needs correctly and understand common sense and context in the way that humans do. And the system needs to ensure that users still feel in control. IoT designers will need to find the right balance of user control and autonomous system actions.

As systems become more complex, platforms need to accommodate automated ways of coordinating devices and software services around user needs. This will require semantic data models and logic. Data models for IoT are in their infancy, but will offer ways of describing devices, system logic, the wider context around devices and activities, and users themselves that can help complex systems coordinate in meaningful and valuable ways.



## [ Appendix A ]

# Companies, Products, and Links

Throughout this book, the authors have discussed a variety of products and companies to illustrate important concepts in and approaches to designing for connected products. The following list of these products and companies, ordered alphabetically, contains relevant links to further information.

PRODUCT	COMPANY	URL
7-segment character layout display	SparkFun	<a href="https://www.sparkfun.com/categories/170">https://www.sparkfun.com/categories/170</a>
Activite, baby monitor, blood pressure cuff, Smart Body Analyzer, baby scale	Withings	<a href="http://www.withings.com">http://www.withings.com</a>
Aether Cone	Aether	<a href="http://www.aether.com">http://www.aether.com</a>
Air Quality Egg	Air Quality Egg	<a href="http://airqualityegg.com">http://airqualityegg.com</a>
Airport Extreme	Apple	<a href="http://store.apple.com/us/product/ME918LL/A/airport-extreme">http://store.apple.com/us/product/ME918LL/A/airport-extreme</a>
Alljoyn	Allseen Alliance	<a href="https://allseenalliance.org">https://allseenalliance.org</a>
Ambient Orb	Ambient Devices	<a href="http://www.ambientdevices.com">http://www.ambientdevices.com</a>
Android	Google	<a href="https://www.android.com">https://www.android.com</a>
Aquaflex Cones Pelvic Floor Exerciser	Savantini	<a href="http://www.aquaflex-cones.co.uk">http://www.aquaflex-cones.co.uk</a>
Automatic	Automatic	<a href="https://www.automatic.com">https://www.automatic.com</a>
Baby monitor	Angelcare	<a href="http://angelcareuk.co.uk/baby-monitors">http://angelcareuk.co.uk/baby-monitors</a>
Beacon	Estimote	<a href="http://estimote.com">http://estimote.com</a>

PRODUCT	COMPANY	URL
Beo4 remote control	Bang & Olufsen	<a href="http://www.bang-olufsen.com/en/picture/remotes/beo4">http://www.bang-olufsen.com/en/picture/remotes/beo4</a>
Bic for Her	Bic	<a href="http://www.bicworld.com/en/products/details/420/bic-for-her">http://www.bicworld.com/en/products/details/420/bic-for-her</a>
BleepBleeps product range	BleepBleeps	<a href="http://bleepbleeps.com">http://bleepbleeps.com</a>
BrewPi	BrewPi	<a href="http://www.brewpi.com">http://www.brewpi.com</a>
Brick	LaCie	Product discontinued, but see <a href="https://www.lacie.com/support/support_manifest.htm?id=10155">https://www.lacie.com/support/support_manifest.htm?id=10155</a>
Bubblino	MCQN Ltd.	<a href="http://bubblino.com">http://bubblino.com</a>
Camera	D-Link	<a href="https://www.mydlink.com/content/productfamily#48tabM99">https://www.mydlink.com/content/productfamily#48tabM99</a>
Chromecast	Google	<a href="http://www.google.com/chrome/devices/chromecast">http://www.google.com/chrome/devices/chromecast</a>
Chune	Clearleft	<a href="http://chune.co.uk">http://chune.co.uk</a>
Cloud bridge	BERGcloud	<a href="http://uk-shop.bergcloud.com/products/berg-cloud-bridge">http://uk-shop.bergcloud.com/products/berg-cloud-bridge</a>
Cloudwash	BERGcloud	<a href="http://bergcloud.com/case-studies/cloudwash">http://bergcloud.com/case-studies/cloudwash</a>
CompuServe	CompuServe, now owned by AOL	Product discontinued, see <a href="http://www.aol.com">www.aol.com</a>
Connected door lock	Danalock	<a href="http://www.danalock.com">http://www.danalock.com</a>
Connected home system	SmartThings	<a href="http://www.smarthings.com">http://www.smarthings.com</a>
Connected home system	Wink	<a href="http://www.wink.com">http://www.wink.com</a>
Crime maps	London Metropolitan Police	<a href="http://maps.met.police.uk">http://maps.met.police.uk</a>
Crock-Pot Smart Slow Cooker with WeMo	Belkin	<a href="http://www.belkin.com/us/p/P-F7C045/">http://www.belkin.com/us/p/P-F7C045/</a>
CST-1 Watch	Central Standard Timing, LLC	<a href="http://centralstandardtiming.com">http://centralstandardtiming.com</a>
Dark Sky	The Dark Sky Company	<a href="http://darkskyapp.com">http://darkskyapp.com</a>

PRODUCT	COMPANY	URL
Dash	Amazon	<a href="https://fresh.amazon.com/dash">https://fresh.amazon.com/dash</a>
Dash button	Amazon	<a href="https://www.amazon.com/oc/dash-button">https://www.amazon.com/oc/dash-button</a>
Digital Life Home Security/Home Automation	AT&T	<a href="https://my-digitallife.att.com/learn">https://my-digitallife.att.com/learn</a>
dLAN 1200+ HomePlug powerline adapter	Devolu	<a href="http://www.devolu.com/en/Products/dLAN-1200+">http://www.devolu.com/en/Products/dLAN-1200+</a>
Door lock	August	<a href="http://august.com">http://august.com</a>
Drivestyle App with Nest	Mercedes-Benz	<a href="http://nest.drivestyleapp.com">http://nest.drivestyleapp.com</a>
Dropcam	Dropcam	<a href="https://www.dropcam.com">https://www.dropcam.com</a>
E2 Engage hub system home energy gateway	Efergy	<a href="http://efergy.com/us/e2-engage-hub-system">http://efergy.com/us/e2-engage-hub-system</a>
Energy efficiency insights	Opower	<a href="http://opower.com">http://opower.com</a>
Evernote	Evernote	<a href="http://evernote.com">http://evernote.com</a>
evohome connected radiator valve	Honeywell	<a href="http://evohome.honeywelluk.com/plan.php">http://evohome.honeywelluk.com/plan.php</a>
Fairphone smartphone	Fairphone	<a href="http://www.fairphone.com">http://www.fairphone.com</a>
Fire phone	Amazon	<a href="http://www.amazon.com/Fire_Phone_13MP-Camera_32GB/dp/B00EOE0WKQ">http://www.amazon.com/Fire_Phone_13MP-Camera_32GB/dp/B00EOE0WKQ</a>
Fire TV	Amazon	<a href="http://www.amazon.com/Fire-TV-streaming-media-player/dp/B00CX5P8FC">http://www.amazon.com/Fire-TV-streaming-media-player/dp/B00CX5P8FC</a>
Fitbit Charge HR	Fitbit	<a href="https://www.fitbit.com/charger">https://www.fitbit.com/charger</a>
Fitbit One	Fitbit	<a href="http://www.fitbit.com/de/one">http://www.fitbit.com/de/one</a>
Galaxy S5 smartphone	Samsung	<a href="http://www.samsung.com/global/microsite/galaxys5">http://www.samsung.com/global/microsite/galaxys5</a>
Glass	Google	<a href="https://www.google.com/glass/start">https://www.google.com/glass/start</a>

PRODUCT	COMPANY	URL
GlowCaps connected pill bottle lid	Vitality	<a href="http://www.vitality.net">http://www.vitality.net</a>
Good Night Lamp	Good Night Lamp	<a href="http://goodnightlamp.com">http://goodnightlamp.com</a>
Google Maps	Google	<a href="http://maps.google.com">http://maps.google.com</a>
GPS Tag	Amber Alert	<a href="http://www.amberalertgps.com">http://www.amberalertgps.com</a>
Hackaball	Made by Many and Map Project Office	<a href="http://www.hackaball.com">http://www.hackaball.com</a>
Heating controller	Tado	<a href="https://www.tado.com/en">https://www.tado.com/en</a>
Hive Active Heating Thermostat	British Gas	<a href="https://www.hivehome.com/hive-active-heating">https://www.hivehome.com/hive-active-heating</a>
Home security system	Scout	<a href="https://www.scoutalarm.com">https://www.scoutalarm.com</a>
HTC One 8 smartphone	HTC	<a href="http://www.htc.com/us/smartphones/htc-one-m8">http://www.htc.com/us/smartphones/htc-one-m8</a>
Hue light bulbs, hub, Hue Tap switch	Philips	<a href="http://www2.meethue.com">http://www2.meethue.com</a>
iBeacon trials	SITA	<a href="http://www.sita.aero/content/Miami-International-and-SITA-cover-airport-beacons">http://www.sita.aero/content/Miami-International-and-SITA-cover-airport-beacons</a>
iConnectHue remote app for Philips Hue	iConnectHue	<a href="http://iconnecthue.com">http://iconnecthue.com</a>
If This Then That	If This Then That	<a href="https://ifttt.com">https://ifttt.com</a>
In-home energy display	Chameleon Technology	<a href="http://www.chameleontechnology.co.uk">http://www.chameleontechnology.co.uk</a>
Instagram	Instagram	<a href="http://instagram.com">http://instagram.com</a>
iPhone/iOS	Apple	<a href="http://www.apple.com/iphone">http://www.apple.com/iphone</a>
iPlayer	BBC	<a href="http://bbc.co.uk/iplayer">http://bbc.co.uk/iplayer</a>
iPod Shuffle	Apple	<a href="http://www.apple.com/ipod-shuffle">http://www.apple.com/ipod-shuffle</a>
Iris home automation system	Lowes	<a href="http://www.lowes.com/cd_iris_239939199_">http://www.lowes.com/cd_iris_239939199_</a>

PRODUCT	COMPANY	URL
Juicy Salif	Alessi	<a href="http://www.alessi.com">http://www.alessi.com</a>
Kegel8 Pelvic Floor Exerciser	Savantini	<a href="http://aquaflex-cones.co.uk">http://aquaflex-cones.co.uk</a>
Kindle	Amazon	<a href="https://kindle.amazon.com">https://kindle.amazon.com</a>
Laguna	Renault	<a href="http://group.renault.com/en/company-vehicles/discover/renault-passenger-vehicles-in-europe">http://group.renault.com/en/company-vehicles/discover/renault-passenger-vehicles-in-europe</a>
Little Printer	BERGcloud	<a href="http://littleprinter.com">http://littleprinter.com</a>
Lockitron door lock	Lockitron	<a href="https://lockitron.com">https://lockitron.com</a>
MacPaint	Apple	Product discontinued, but see <a href="http://en.wikipedia.org/wiki/MacPaint">http://en.wikipedia.org/wiki/MacPaint</a>
MagicBand	Disney	<a href="https://disneyworld.disney.go.com/plan/my-disney-experience/bands-cards">https://disneyworld.disney.go.com/plan/my-disney-experience/bands-cards</a>
Microchip cat door	SureFlap	<a href="https://www.sureflap.com/en-us/pet-doors/microchip-cat-flap">https://www.sureflap.com/en-us/pet-doors/microchip-cat-flap</a>
Microcontrollers	Arduino	<a href="http://www.arduino.cc">http://www.arduino.cc</a>
MilsCard cryptographic security module	Mils Electronic	<a href="http://www.mils.com/en/technology/security-tokens/#1">http://www.mils.com/en/technology/security-tokens/#1</a>
Mimo baby monitor	Rest Devices, Inc.	<a href="http://mimobaby.com">http://mimobaby.com</a>
Minority Report	Twentieth Century Fox Film Corporation	<a href="http://www.imdb.com/title/tt0181689">http://www.imdb.com/title/tt0181689</a>
Movement monitor and bed sensor	Emfit	<a href="http://www.emfit.com/en/care/products_care/movement-monitor">http://www.emfit.com/en/care/products_care/movement-monitor</a>
Multi Clock	Muji	Product discontinued, but see <a href="http://www.muji.com">http://www.muji.com</a> and <a href="http://www.coolhunting.com/design/muji_multifunct">http://www.coolhunting.com/design/muji_multifunct</a>
My Energy Report	British Gas	<a href="http://www.britishgas.co.uk/smarter-living/control-energy/smart-meters/my-energy-use.html">http://www.britishgas.co.uk/smarter-living/control-energy/smart-meters/my-energy-use.html</a>
MyHome home automation system	Control4	<a href="http://www.control4.com">http://www.control4.com</a>

PRODUCT	COMPANY	URL
Nabaztag	Violet (now defunct)	Information available at <a href="http://en.wikipedia.org/wiki/Nabaztag">http://en.wikipedia.org/wiki/Nabaztag</a>
Nest thermostat	Nest	<a href="https://nest.com/thermostat/life-with-nest-thermostat">https://nest.com/thermostat/life-with-nest-thermostat</a>
Netflix	Netflix	<a href="https://www.netflix.com">https://www.netflix.com</a>
Nexus Q	Google	Product discontinued, but see <a href="http://en.wikipedia.org/wiki/Nexus_Q">http://en.wikipedia.org/wiki/Nexus_Q</a>
Nike+	Nike	<a href="http://nikeplus.nike.com/plus">http://nikeplus.nike.com/plus</a>
Ninja Sphere gateway	Ninja Blocks	<a href="https://ninjablocks.com">https://ninjablocks.com</a>
NodeRED	IBM Emerging Technology	<a href="http://nodered.org">http://nodered.org</a>
Octopus card	Octopus Holdings, Ltd.	<a href="http://www.octopus.com.hk">http://www.octopus.com.hk</a>
OLPC laptop	One Laptop Per Child	<a href="http://one.laptop.org">http://one.laptop.org</a>
OpenHAB	OpenHAB UG	<a href="http://www.openhab.org">http://www.openhab.org</a>
Organic matter sensor	Lapka	<a href="https://mylapka.com/pem">https://mylapka.com/pem</a>
Oyster card	Transport for London	<a href="https://account.tfl.gov.uk/oyster?App=8ead5cf4-4624-4389-b90c-b1fd1937bfff&amp;returnURL=https://oyster.tfl.gov.uk/oyster/security_check">https://account.tfl.gov.uk/oyster?App=8ead5cf4-4624-4389-b90c-b1fd1937bfff&amp;returnURL=https://oyster.tfl.gov.uk/oyster/security_check</a>
Parker	Streetline	<a href="http://www.streetline.com/find-parking/parker-mobile">http://www.streetline.com/find-parking/parker-mobile</a>
Pebble Smartwatch	Pebble	<a href="http://getpebble.com">http://getpebble.com</a>
Pet GPS tracker	Tractive	<a href="https://tractive.com">https://tractive.com</a>
PFX2V Pelvic Floor Exerciser with Biofeedback	Laborie	<a href="http://www.laborie.com/products/pelvic-floor-rehabilitation/pfx.html">http://www.laborie.com/products/pelvic-floor-rehabilitation/pfx.html</a>
Protect smoke/CO alarm	Nest	<a href="https://nest.com/smoke-co-alarm/life-with-nest-protect">https://nest.com/smoke-co-alarm/life-with-nest-protect</a>
Protosnap	LilyPad	<a href="https://www.sparkfun.com/products/11262">https://www.sparkfun.com/products/11262</a>

PRODUCT	COMPANY	URL
Quirky+GE Pivot Power Genius	Quirky	<a href="https://www.quirky.com/shop/633">https://www.quirky.com/shop/633</a>
Reactable	Reactable	<a href="http://www.reactable.com">http://www.reactable.com</a>
Revolv home automation system	Revolv (now owned by Nest)	<a href="http://revolv.com">http://revolv.com</a>
Roomba 870 vacuum cleaner	iRobot Corporation	<a href="http://www.irobot.com">http://www.irobot.com</a>
Sbeam	Samsung	<a href="http://www.samsung.com/global/galaxy3/sbeam.html">http://www.samsung.com/global/galaxy3/sbeam.html</a>
Scratch	MIT	<a href="https://scratch.mit.edu">https://scratch.mit.edu</a>
Screenfridge	Electrolux	<a href="http://www.electroluxgroup.com/en/history-1990-1999-764">http://www.electroluxgroup.com/en/history-1990-1999-764</a>
SCX-3405 laser printer	Samsung	<a href="http://www.samsung.com/hk_en/consumer/computer-peripherals/printers-multifunction/monochrome-laser-multifunction-printers-faxes/SCX-3405/XSS">http://www.samsung.com/hk_en/consumer/computer-peripherals/printers-multifunction/monochrome-laser-multifunction-printers-faxes/SCX-3405/XSS</a>
SenseAware	Fedex	<a href="http://www.senseaware.com">http://www.senseaware.com</a>
Shine	Misfit Wearables	<a href="http://misfit.com/products/shine">http://misfit.com/products/shine</a>
Shodan search engine	Shodan	<a href="https://www.shodan.io">https://www.shodan.io</a>
Sifteo Cubes	Sifteo	Product discontinued, but see <a href="https://www.sifteo.com/home">https://www.sifteo.com/home</a>
SimpleWave for VERA	SimpleWave	<a href="http://www.simplewave.com">http://www.simplewave.com</a>
skydrop smart irrigation controller	skydrop	<a href="http://www.skydrop.com">http://www.skydrop.com</a>
Sleep Monitor	Beddit	<a href="http://www.beddit.com">http://www.beddit.com</a>
Smart meters	British Gas	<a href="http://www.britishgas.co.uk/business/energy-services/monitoring/smart-metering">http://www.britishgas.co.uk/business/energy-services/monitoring/smart-metering</a>
Smart meters	Enexis	<a href="https://www.enexis.nl">https://www.enexis.nl</a>
Smart pill	Proteus Health	<a href="http://www.proteus.com">http://www.proteus.com</a>
Smart vending machine	J R East Water Business	<a href="http://www.acure-fun.net">http://www.acure-fun.net</a>

PRODUCT	COMPANY	URL
Smartplug, home monitoring kit	AlertMe	<a href="http://www.alertme.com">http://www.alertme.com</a>
Spotify	Spotify	<a href="https://www.spotify.com">https://www.spotify.com</a>
Staples Connect hub	Staples	<a href="http://www.staples.com/sbd/cre/marketing/staples-connect">http://www.staples.com/sbd/cre/marketing/staples-connect</a>
Thingful	Thingful	<a href="https://thingful.net">https://thingful.net</a>
Tile	Tile	<a href="https://www.thetileapp.com">https://www.thetileapp.com</a>
Time It Right home automation system	Autotime LLC	<a href="http://www.autotimeonline.com">http://www.autotimeonline.com</a>
Touch	3D Systems	<a href="http://www.geomagic.com/en/products/sculpt/overview">http://www.geomagic.com/en/products/sculpt/overview</a>
TouchID fingerprint sensing	Apple	<a href="http://www.apple.com/iphone-6/touch-id">http://www.apple.com/iphone-6/touch-id</a>
Toymail Mailman (Snort)	Toymail Co.	<a href="http://www.toymail.co">http://www.toymail.co</a>
Tracklight	SunSprite	<a href="https://www.sunsprite.com">https://www.sunsprite.com</a>
Two step verification	Google	<a href="https://www.google.com/landing/2step">https://www.google.com/landing/2step</a>
Ugle	Voy	<a href="http://www.voyoslo.com/projects/ugle">http://www.voyoslo.com/projects/ugle</a>
Up	Jawbone	<a href="https://jawbone.com/up">https://jawbone.com/up</a>
Vantage washer and dryer	Whirlpool	<a href="http://www.whirlpool.com/-[WTW7990XG]-1001711/WTW7990XG">http://www.whirlpool.com/-[WTW7990XG]-1001711/WTW7990XG</a>
Vodafone live!	Vodaphone	<a href="http://vodafone.com">http://vodafone.com</a>
Water shutoff valve	Fortrezz	<a href="http://www.fortrezz.com/index.php/products/water-valve">http://www.fortrezz.com/index.php/products/water-valve</a>
Weather Station	NetAtmo	<a href="https://www.netatmo.com/en-US/site">https://www.netatmo.com/en-US/site</a>
WeMo switch smart plug	Belkin	<a href="http://www.belkin.com/us/F7C027-Belkin/p/P-F7C027">http://www.belkin.com/us/F7C027-Belkin/p/P-F7C027</a>
Whiskey gifting	Diageo/EVRYTHNG	<a href="https://evrythng.com/customers/diageo-case-study">https://evrythng.com/customers/diageo-case-study</a>
WigWag	WigWag	<a href="http://www.wigwag.com">http://www.wigwag.com</a>



PRODUCT	COMPANY	URL
Wii Remote	Nintendo	<a href="http://www.nintendo.com/wiiu/accessories">http://www.nintendo.com/wiiu/accessories</a>
Windows Phone 8	Microsoft	<a href="http://www.windowsphone.com">http://www.windowsphone.com</a>
WW9000 connected washing machine	Samsung	<a href="http://www.samsung.com/uk/consumer/home-appliances/laundry/washing-machines/WW10H9600EX/EU?subsubtype=eco-bubble-vrttm">http://www.samsung.com/uk/consumer/home-appliances/laundry/washing-machines/WW10H9600EX/EU?subsubtype=eco-bubble-vrttm</a>
X10 Powerhouse Commodore 64	Commodore/X10	Product discontinued, but see <a href="http://www.commodore.ca">http://www.commodore.ca</a>
Xbox	Microsoft	<a href="http://www.xbox.com">http://www.xbox.com</a>
XFINITY home security and automation system	Comcast	<a href="http://www.comcast.com/home-security.html">http://www.comcast.com/home-security.html</a>
Jaguar XF dashboard	Jaguar Landrover	<a href="http://www.jaguar.com/index.html">http://www.jaguar.com/index.html</a>
Zenius coffee machine	Nespresso	<a href="https://www.nespresso.com/pro/uk/en/product/zenius">https://www.nespresso.com/pro/uk/en/product/zenius</a>
Zipcar	Zipcar	<a href="http://www.zipcar.com">http://www.zipcar.com</a>
Zite.com	Flipboard	<a href="http://zite.com">http://zite.com</a>



# [ Index ]

## Symbols

3D modeling/rendering, 248–249  
3D printing, 252, 270–271  
6LOWPAN project, 402

## A

Abrams, Rachel, 268  
accelerometers, 50–51  
access control feature (home automation systems), 13  
accessibility of embedded devices, 321–322  
access points, 499–500  
actionable insights from data, 554–556  
active sensors, 48  
activity feeds, 493–495  
actors and stakeholders, 154–162  
actuators, 52–54  
Adaptive Path consultancy, 194  
addressability of devices on Internet, 65  
AEIOU schema, 177  
aesthetic styling across platforms, 364–366  
Aether Cone connected speaker, 280–281, 306–307  
affordance (physical devices), 224–225, 258–259  
aggregation of data, 442–444  
Air Quality Egg map of sensors, 401  
air quality sensors, 50, 533  
alarm systems, 143  
AlertMe  
    home monitoring kit, 471–472  
    smartplug, 74  
    system for pairing devices, 511  
alerts, managing, 482–483  
AllJoyn services, 394, 395

Amazon  
    Dash, 232, 292  
    Fire Phone, 308–309  
    Fire TV remote, 319–320  
Amber Alert GPS tags, 439  
ambient analytics systems, 547  
Ambient Devices, 318  
ambient interfaces, 317–318  
Ambient Orb, 282–283  
Ambient Umbrella, 283  
Amit, Gadi, 260–261  
Android Beam (NFC), 45, 84  
Android devices sharing WiFi details over NFC, 507  
Android unlock pattern, 421  
Angelcare baby monitor, 82  
anthropometrics, 257–259  
APIs (application programming interfaces)  
    connecting Internet services with, 70–71  
    granularity design issues, 96–98  
    integrating Internet services with, 397–399  
    overview, 94–95  
    structure of, 98–100  
    third-party, 100  
    web API basics, 95–96  
appearance models, 252  
Apple  
    iBeacons, 347  
    TouchID fingerprint sensing, 422  
    TV, 319  
    VoiceOver, 322  
application layer (networking), 77  
application protocols, 91–92  
Arduino microcontrollers, 37  
Arieff, Allison, 458  
Arup, 584  
Ashton, Kevin, xvi

- aspirations
    - defining with portraits, 190
    - goals and (product values), 187
  - AT&T Digital Life tablet app, 131
  - audio/voice interfaces, 290–294
  - August door lock, 131
  - authentication (security), 421–423, 432–434
  - authorization (security), 422
  - automatic gearboxes, 346–347
  - automation
    - Automation by XKCD, 645
    - and context-insight framework (data), 545–547
    - designing for, 499
    - programming, 11–12
    - pros and cons of, 211–212
  - autonomous systems
    - balancing user control with, 652–653
    - risks of, 650–651
  - availability (security), 420
- B**
- baby monitors, 82–83
  - barcodes
    - computer vision and, 311–312
    - two-dimensional (QR codes), 45
  - batteries, conserving life of, 53–56
  - BBC iPlayer, 1
  - beacons. *See also* iBeacons; Estimote beacons; Physical Web project
    - airline trials with, 121
    - device pairing and, 514
    - overview, 46–48
  - Beart, Pilgrim, 86, 471–472
  - behavioral changes
    - feedback loops, 567
    - goal of, 565–568
    - persuasion and, 208
  - behavioral context (research), 164, 169
  - behavioral design (physical devices), 218, 226
  - behavior, patterns of (privacy), 445–446
  - Belkin
    - WeMo app, 490, 613
    - WeMo smart plug, 125–126
    - WeMo Switch, 69, 231, 372
  - Bellotti, Victoria, 448
  - Beo4 TV remote control, 266
  - BERG
    - Cloud Bridge, , 260
    - Little Printer, 239–240, 260, 265
    - prototyping Connbox, 578–581
    - washing machine app, 61
  - BiC for Her pens, 156
  - big data, 537, 542–543
  - bill of materials (BOM), 272–273
  - Bishop, Durrell, 297
  - Blackwell, Alan, 13
  - Blank, Steve, 120
  - BleepBleeps products, 262
  - Blink(1) notification light, 583
  - BlinkUp, 506
  - blue collar coders, 143
  - Bluespray garden irrigation controller, 354
  - Bluetooth
    - devices enabled for, 46
    - numeric comparison pairing, 516
    - OOB pairing via NFC, 517
    - overview, 80–82
    - pairing options, 513–515
  - Botanicalls wireless moisture sensor, 207
  - Bowles, Cennydd, 305
  - brand image/promise (physical devices), 262–263
  - Brand, Stewart, 133
  - Bransfield, Gene, 425
  - Braungart, Michael, 462, 463
  - BRCK
    - iterative router design, 576–577
    - mobile WiFi router (case study), 519–526
    - router testing, 163–164
  - BrewPi fermentation temperature controller, 126
  - Brick stackable external hard drive, 229
  - bridges. *See also* gateways
    - user interaction with, 228–229
  - Brignull, Harry, 477
  - British Gas
    - dual fuel in-home display, 139
    - Hive Active Heating controller, 353, 362
    - Hive app, 305
    - Hive app homescreen, 494
    - Hive system, 352

Hive thermostat, 36  
smart meter report, 554  
Bubblino Internet-connected bubble  
machine, 53  
Buchenau, Marion, 591  
business models, 144–148  
Business Origami, 595–596  
Buxton, Bill, 183, 606

## C

CAD (computer-aided design),  
250–251  
CAM (computer-aided  
manufacturing), 250–251  
canned (archived) data (prototyping),  
598  
car communications/entertainment  
connectivity system,  
327–336  
caregiving (Proteus connected pill),  
106–107  
car navigation systems, 535  
case studies  
BRCK mobile WiFi device,  
519–526  
Ford SYNC3, 327–336  
Little Kelham, 215–220  
LOOP pelvic floor exerciser,  
413–418  
Proteus connected pill, 103–109  
Cassimally, Hakim, xix  
CE (Conformité Européene)  
markings, 254  
cellular data connectivity, 79–80  
character set displays, 285–287  
Charlier, Martin, 221, 275, 467  
Chromecast (Google), 235  
Chune speaker prototype, 256  
Citrix Worx storyboard, 587  
Clearfield, Chris, 426  
Clearleft design studio, 255–256  
closed system microcontrollers, 37  
Clouds, Edges, Fog, and the  
Pendulum of Distributed  
Computing, 537  
cloud servers vs. edge devices, 536  
Cloudwash washing machine  
(BERG), 315  
CMF (color/material/finish)  
research, 242  
CNC (computer numerical  
controlled) machines,  
250–251  
coarse-grained APIs, 97  
co-design workshops, 179  
coding (programming), 632  
cognitive mapping (research),  
172–173  
collages, prototyping with, 582–584  
Comcast XFINITY Home, 636  
common functional components  
(platforms), 496–498  
companies, list of, 667–675  
complex systems  
basics, 612–617  
controlling. *See* controlling  
complex systems  
data models, 656–666  
machine-control of, 646–653  
scaling platforms, 653–656  
scaling the UX. *See* scaling the UX  
composition  
determining right, 357–360  
patterns of, 350–358  
CompuServe of Things, 382–385  
computers, multipurpose, 31  
computer vision, 308, 311–312  
conceptual models (IoT design)  
domain metaphors and, 195–198  
overview, 26–27  
user/design models, 341–351  
confidentiality, 420  
connected car systems, 327–336  
connected devices  
actuators, 52–54  
connected sensors, 39–42  
conserving battery life, 53  
disposal of, 461–462  
embedded devices, 32–36  
embedded hardware, 36–37  
embedded software, 37–39  
flow of data in, 533–537  
levels of interactivity of, 313–315  
multipurpose computers, 31  
overview, 29–30  
passively trackable objects, 42–48  
powering, 53–56  
sending commands between, 493  
sensors, 48–53  
types of, 30–31  
connected lighting systems, 344–345

- connected products research
    - techniques
      - cognitive mapping, 172–173
      - diaries/activity logs, 175–176
      - field visits, 176–178
      - generative methods, 178–180
      - LEGO models, 179
      - overview, 169–171
      - spatial mapping, 171–172
      - timelines, 172–174
      - translating research into product definitions. *See* product definitions
      - verbal explanations, 176–177
  - connected sensors, 39–42
  - connectivity issues in IoT systems, 10
  - consistency across multiple devices
    - aesthetic styling, 364–366
    - device combinations, 368
    - following platform conventions, 362–364
    - interaction architecture and functionality, 366–367
    - overview, 360–361
    - terminology, 361–362
  - construction geometry (product manufacturing), 271
  - consumers. *See also* users
    - convincing, 118–119
    - innovators vs., 116–119
  - contactless ticketing and payments, 44–45
  - context
    - capturing contextual information, 623–624
    - context-aware computing, 304–305
    - context-insight/automation framework (data), 545–547
    - context of interaction (connected products), 162
    - context-sensitive interactions, 304–308
    - evaluating data relative to, 552–554
  - contextual integrity, 440
  - continuity in cross-platform design
    - cross-device interaction/task migration, 376–380
    - data/content synchronization, 368–375
    - overview, 368
    - spotting continuity errors, 368
  - continuous iteration (product design), 238–240
  - Control4 MyHome, 640
  - controlling complex systems
    - displacement of actions from results, 629–631
    - by machines, 646–653
    - overview, 624–626
    - by user programming, 626–629, 632–647
  - controls, designing
    - activity feeds, 493–495
    - grouping controls, 492
    - overview, 487–488
    - prioritizing key controls, 488–491
    - sending commands between devices, 493
  - Coraggio, Giulio, 447
  - Cradle to Cradle, 462
  - Crock-Pot Slow Cooker with Wemo, 127, 248–249
  - Croll, Alistair, 537
  - cross-device interactions, 376–380
  - Crossing the Chasm, 116
  - cross-media systems, 357–358
  - cross-platform UX
    - composition
      - determining right, 357–360
      - patterns of, 350–358
    - conceptual models, 341–351
    - consistency across multiple devices. *See* consistency across multiple devices
    - continuity. *See* continuity in cross-platform design
    - interusability, 340–341
    - overview, 338–340
  - CST-1 watch, 287
  - curvature combs (3D modeling), 249
  - customer journey maps, 193–196
  - custom segment displays, 284–287
- D**
- Daily Me news UX, 451
  - Danalock connected door lock, 52
  - Dark Sky weather app, 556
  - Dash, Anil, 143
  - dashboards, 489–491
  - data
    - aggregation of, 442–444

- big data vs. small data, 543
  - collecting/distributing, 440–441
  - data layer (interoperability), 385–386
  - data models (complex systems), 656–666
  - data ownership (product definitions), 209–211
  - data points, missing, 41
  - data science, 537
  - dynamic vs. static, 540
  - handling with tact, 437–440
  - inferred vs. direct, 540–541
  - insights from, 554–556
  - interoperable, 393
  - protection of, 445–448
  - prototyping with, 596–599
  - user, 665
  - data-driven design, 573
    - behavioral change, 565–568
    - data from multiple products, 549
    - data science, 537–538
    - experience over time, 558–559
    - flow of data in connected products, 533–537
    - inferences and interpretation, 561–562
    - making data meaningful/actionable, 550–556
    - objectivity of data, 564–565
    - overview, 16–17, 527–528
    - qualitative data and, 572
    - respecting user attention span, 558–561
    - third-party data, 544–545
    - transparency and trust, 563–564
    - types of data-driven products, 544–549
    - types of IoT data, 538–542
    - using data in design process, 572
    - visualizing data, 569–571
  - Davies, Russell, 505
  - debugging (programming), 633
  - dedicated gateways, 66–68
  - demographic statistics, stereotyping based on, 155–157
  - Denis, Charles, 340
  - design. *See also* IoT design models for assembly (DFA), 253
    - challenges of, xvii–xviii
    - design freeze, 238–239
    - facets of, 20–21
    - industrial. *See* industrial design for manufacturability (DFM), 253–254
    - principles of, 199
    - and user models, 341–343
  - Devenish, Ryan, 201
  - devices
    - connected. *See* connected devices
    - device abstraction, 402–403
    - device-to-device connections, 69–70
    - disposal of, 461–462
    - management of, 480–483
    - mobile. *See* mobile devices
    - pairing, 509–513. *See also* pairing devices
    - system-generated metadata about, 658–661
    - upgradeability of, 461
  - Devolo dLAN 1200+ HomePlug powerline adapter, 84
  - DFA (design for assembly), 253
  - DFM (design for manufacturability), 253–254
  - Diageo whisky gifting campaign, 46
  - diaries/activity logs (research), 175–176
  - direct manipulation concept, 11–12, 626–628
  - direct vs. inferred data, 540–541
  - discovering presence of systems, 486–487
  - Disney Magic Band, 44, 531, 548–549
  - Disney theme parks, 44
  - displacement of actions from results, 629–631
  - displays, prototyping, 602–603
  - disposal of devices, 461–462
  - D-Link camera app, 408–409
  - domain metaphors (product definitions), 195–198
  - domain models, system, 663–665
  - Dropcam connected camera, 49, 81
  - Duhigg, Charles, 186
  - dynamic displays, 288
  - dynamic vs. static data, 540
- E**
- early adopters, 117, 190
  - early majority users, 117
  - ecological context (research), 167–169

- Economizer home electricity
    - monitoring video, 590
  - ecosystems
    - maps, 205, 206
    - service, 140–143
  - edge devices. *See also* embedded devices
    - cloud servers vs., 536
    - connecting to Internet, 499–508
    - connecting to one another, 69–70
    - connecting via cloud platforms, 397
    - connecting via local gateways/hubs, 395–397
    - coordinating, 407–408
    - definition of, 30
    - determining compatibility of, 407
    - directly connecting, 394–396
    - IP communications to, 85–86
    - keeping simple, 72
    - organizing/accessing controls, 409–411
  - Edwards, Gawain, 482, 488, 614
  - Efergy energy monitoring service, 130
  - Ehrnberger, Karin, 263–264
  - Electric Imp, 37, 310, 505
  - Electrolux Screenfridge, 152–153
  - electronic ink displays, 288–289
  - Elements of User Experience diagrams, 19
  - elevator pitch, 184–185
  - embedded devices, 32–36. *See also* edge devices; physical device design
  - embedded hardware, 36–37
  - embedded software, 37–39
  - emotional design, cycle of, 335
  - Empathy Belly, 192–193
  - Enabled by Design, 321–322
  - encryption, public key, 422
  - energy
    - measuring consumption of, 553–554
    - usage of connected products, 460–461
  - engineering/production (industrial design), 252–254
  - environment
    - impact of IoT products, 459–463
    - overview, 457–459
    - role of designers in protecting, 463–464
  - ergonomics, 257–259
  - Ericsson User Experience Labs, 592–594
  - escape holes (product manufacturing), 271
  - Estimote beacons, 47, 261
  - Ethernet connectivity (networking), 78
  - European privacy laws, 446–447
  - Evans, Benedict, xv
  - event logs, 493–495
  - Evernote, 339
  - Evolve Collaborative, 242–243
  - E-waste recycling, 462
  - experience maps, 193–196
  - experience prototypes, 591–592
  - Experientia design consultancy, 180
- F**
- face recognition/detection, 308
  - facets of design, 20–21
  - Fairphone smart phone, 464
  - faking it (Wizard-of-Oz prototyping), 598–599
  - FCC Declaration of Conformity, 254
  - FedEx SenseAware service, 132–133, 594
  - feedback loop model (behavior), 567, 567–568
  - “feels like” prototypes, 606
  - Ficklin, Jared, 312
  - field visits (research), 176–178
  - fine-grained APIs, 96–97
  - firmware, 37
  - Fitbit activity tracker, 288
  - fog computing (Cisco), 537–538
  - force feedback mechanisms, 302–303
  - Ford SYNC3 (case study), 327–336
  - formulating design language, 245–247
  - Fortrezz water shutoff valve, 52
  - frequency of data readings, 543
  - future routers project, 235–236
- G**
- Galaxy S5 heart rate sensor, 49
  - Gale, Peter, 477
  - Garrett, Jesse James, 19
  - Gasser, Urs, 385



- gateways
    - advantages of, 72–73
    - basics of, 66
    - connecting edge devices via, 395–397
    - dedicated, 66–68
    - disadvantages of, 73–75
    - setting up, 499–508
    - smartphones as, 68–69
    - user interaction with, 228–229
  - gearboxes, automatic, 346–347
  - generalizing among people groups (product definitions), 190
  - generative methods (research), 178–180
  - gestural interactions, 294–297
  - GET method (web APIs), 96
  - ghosting effect, 289
  - glanceable interfaces, 317–318
  - GlowCaps connected pill bottle, 6, 79, 204–205
  - goals and aspirations (product values), 187
  - goals and motivations of users, 551–552
  - Goldilocks interface, 218
  - Good Grips brand (OXO), 322
  - Goodman, Elizabeth, 151, 183, 575
  - Good Night Lamp, 508
  - Google
    - Contacts (push notifications), 90–91
    - Glass, 166–167, 437–441
    - Maps, 404
  - GPRS radio transmissions, 79–80
  - GPS
    - receivers/satellites, 50
    - tracking units, 456
  - granularity design issues (APIs), 96–98
  - Greenfield, Adam, xix
  - grounded innovation (product definitions), 198–199
  - grouping controls, 492
  - gyroscopes, 50
- H**
- Hackaball programmable ball, 22, 24
  - hackathons (Enabled by Design), 321–322
  - Hall, Erika, 182
  - Hapifork, 531
  - hardware
    - cryptographic security module, 431
    - embedded, 36–37
    - engineering, 252–253
  - Havaianas accessories, 202
  - Hawthorn effect, 107
  - HCI (human-computer interaction), 1
  - headless Ethernet/WiFi devices, 503
  - headless gateways, 499–500
  - heart rate monitors, 570
  - heart rate sensors (Galaxy S5), 49
  - heating controllers, 370
  - heating controller with modal functions, 367
  - heating service (composition), 352–354
  - Heinrich, Louisa, 663
  - Hemmert, Fabian, 303
  - Heuristic Evaluation (Usability Inspection Methods), 360
  - Hill, Dan, 584
  - historical vs. real-time data, 542–543
  - The Hitchhiker’s Guide to the Galaxy, 649
  - home automation system
    - Home Automation profile, 511
    - home platform, 14
    - market, 118
    - SmartThings ecosystem, 339
    - system architecture, 67
  - HomeOS, 642
  - HomePlug standard, 84
  - home security systems, 168, 208, 664–665
  - home systems setups. *See also* programming, user
    - Little Kelham, 215–220
    - progressing complexity of, 614–617
    - user needs and, 621–623
  - Honan, Matt, 166
  - Honeywell evohome connected radiator valve, 5
  - Hong Kong Octopus travel cards, 6–7
  - HTTP (HyperText Transport Protocol), 77, 91
  - Hubert, Elisabeth, 3
  - hubs. *See also* gateways
    - user interaction with, 228–229

- Hue (Philips)
    - Bridge, 230
    - bulbs, 232–233
    - lighting app, 408
    - Tap remote control, 234–235
  - human-computer interaction (HCI), 1
  - Human Interface Device profile, 511
  - human layer (interoperability), 386
  - Hunch.com, 444
  - Hussain, Sofia, 205, 207
  - Huuskonen, Pertti, 147
  - HyperCat standard, 405
  - hypertension treatment (Proteus connected pill), 107
- I**
- iBeacons (Apple), 122, 347
  - IDA Institute (website), 21
  - If This Then That functionality, 115, 125, 635
  - Igoe, Tom, 276
  - immersion tools (product values), 191–195
  - industrial design
    - CAD/CAM, 250–251
    - CMF (color/material/finish) research, 242
    - collaboration with UX, 255–257
    - detailing/developing designs, 248–250
    - engineering/production, 252–254
    - formulating design language, 245–247
    - mindset of, 238–239
    - mood boards (design themes), 242–243
    - overview, 23–24
    - of physical hardware, 18–19
    - sketching, 243–246
  - inferences/interpretation of data, 561–562
  - inferred vs. direct data, 540–541
  - information overload, 558–561
  - informed consent (privacy), 448
  - ingestible sensor (Proteus), 104
  - innovation to mass market
    - convincing consumers, 118–119
    - innovators vs. consumers, 116–119
    - types of product markets, 121–123
    - value propositions for IoT, 120–123
  - input-only devices, 232
  - insights from data, 554–556
  - inspiration, sources of (product definitions), 190
  - Instagram mobile app, 371–372
  - institutional layer (interoperability), 386
  - instructions (system setups), 474–478
  - Insure the Box car insurance, 148
  - intangible interactions, 296–297
  - integrity, definition of, 420
  - intelligent systems
    - balancing user control with autonomous, 652–653
    - capabilities of, 647–650
    - risks of autonomous, 650–651
  - interaction design (IoT), 22–23
  - interactions, key. *See* key interactions
  - interactions with physical devices
    - accessibility issues, 321–322
    - ambient interfaces, 317–318
    - audio/voice interfaces, 290–294
    - context-sensitive interactions, 304–308
    - gestural interactions, 294–297
    - glanceable interfaces, 317–318
    - interface combinations, 312
    - interfaces with limited input/output, 318–320
    - levels of interactivity, 313–315
    - mobile and web interfaces, 315–317
    - multimodal interactions, 312
    - physical controls, 278–282
    - placement relationship to, 228–238
    - seeing devices, 308–312
    - tactile interactions, 301–304
    - tangible interactions, 297–302
    - types of, 276–278
    - universal design principles, 322–324
    - visual/screen interfaces, 282–291
  - Internet
    - addressability on, 65
    - connecting devices to, 499–508
    - direct connections to devices, 69–70
    - Internet networking basics, 76–77
    - Internet Protocol (IP), 76

- networking connectivity options, 77–80
- service to service via, 397–399
- unreliability of, 7
- Internet services
  - APIs. *See* APIs (application programming interfaces)
  - productization and, 115–116
  - role of, 93–94
- interoperability
  - basics, 385–388
  - between products and services, 399
  - challenges of, 14–16
  - CompuServe of Things, 382–385
  - connecting edge devices via cloud platforms, 397
  - connecting edge devices via local gateways/hubs, 395–397
  - connecting gateways to each other, 397
  - determining device compatibility, 407
  - directly connecting edge devices, 394–396
  - enabling with gateways, 73
  - functioning together vs. true, 400–401
  - generic vs. specific UIs, 408–409
  - getting devices to coordinate, 407–408
  - improving, 401–405
  - interoperable data, 393
  - network interoperability, 65, 388–393
  - organizing devices/accessing controls, 409–411
  - service to service via Internet, 397–399
  - user control and, 411
- interusability, 23, 379–380, –
- inventory tracking (RFIDs), 42
- Ionescu, Arna, 103
- IoT architecture
  - advantages of gateways, 72–73
  - dedicated gateways, 66–68
  - device-to-device connections, 69–70
  - direct Internet connections, 69–70
  - disadvantages of gateways, 73–75
  - effects on UX, 71–72
  - overview, 66
  - service-to-service connections, 70–71
  - smartphones as gateways, 68–69
- IoT design models
  - conceptual model, 26–27
  - facets of design, 20–21
  - industrial design. *See* industrial design
  - interaction design, 22–23
  - interusability, 23
  - overview, 17
  - platform design, 26–27
  - productization, 26
  - service design, 24–25
  - UI/visual design, 18, 21
- IoT devices
  - addressability on Internet, 65
  - asynchronous nature of, 8–10
  - basics of, 2–3
  - business models for, 145–146
  - combining with services, 135
  - designs showing state of, 62–63
  - device-to-device connections, 69–70
  - distribution in real world, 11
  - intermittent network connections for, 59
  - programming, 11–12
  - user understanding of, 136–140
  - variety of, 5–6
- IoT UX
  - networking relevance to, 58–59
  - overview, 1–4
- IoT UX vs. digital services UX
  - asynchronous nature of IoT, 8–10
  - complexity of services, 13
  - data-based design, 16–17
  - device distribution in real world, 11
  - focus on cloud service, 6–8
  - functionality across multiple devices, 5–6
  - Internet unreliability, 7
  - interoperability challenges, 14–16
  - programming-like activities, 11–12
  - vulnerability of code running in many places, 9–11
- IP communications to edge devices, 85–86

- iPhones
    - Bluetooth pairing, 515
    - passkey entry pairing, 516
  - iPods
    - bathtub comparison, 246
    - iPod Shuffle, 223
  - IPSO Alliance, 658–660
  - iRobot Roomba 870 vacuum cleaner, 646
  - iterative design
    - necessity of, 576–577
    - prototyping. *See* prototyping
    - UX evaluation, 607–608
  - Ive, Jonathan, 253
- J**
- Jawbone UP, 534
  - Jenson, Scott, 134, 648, 46
  - Johansson, Victor, 255
  - Johnson, Mark, 196
  - Jones, Matt, 580, 652–653
  - journey maps, 193–196
  - Juicy Salif lemon squeezer, 226
  - “just works” approach (device pairing), 514–515
- K**
- Kare, Susan, 12
  - Karsenty, Laurent, 340
  - Karten Design consultancy, 190
  - Kawano, Mark, 142
  - key interactions
    - connecting devices to Internet, 499–508
    - designing controls. *See* controls, designing
    - device management, 480–483
    - discovering presence of systems, 486–487
    - notifications/messages/alerts, 482–483
    - overview, 467–468
    - pairing devices. *See* pairing devices
    - platforms, 494–498
    - setting up mobile devices, 507–508
    - setting up systems. *See* setting up systems
    - system maintenance, 485–486
    - user management, 483–485
  - kilowatt-hours (kWh), 553
  - Kindle Whispersync, 369
  - Kinect Sports 2 on Xbox, 626–627
  - Korea Telecom products, 246–247
  - Koster, Michael, 656
  - Krippendorff, Klaus, 176
  - Kruzeniski, Mike, 167
  - Kuniavsky, Mike, xix, 7, 17, 137, 197
- L**
- Lakoff, George, 196
  - Lanclos, Donna, 172–173
  - languages, common design, 497
  - Lapka organic matter sensor, 458
  - late majority users, 117
  - latency
    - and responsiveness, variations in (networking), 60–62
    - controlling over local networks, 73
  - Law of Conservation of Complexity, 348
  - LED light interfaces, 282–285
  - levers, value, 188
  - LG smart TVs, 563
  - LIFX lighting, 14, 323, 398, 406, 407
  - Light, Ann, 348, 419
  - lighting systems, 344–345, 497
  - light output interfaces, 282–285
  - LilyPad Protosnap wearables package, 600–601
  - Lim, Youn-Kyung, 581
  - link layer protocols (networking), 76
  - Little Kelham (case study), 215–220
  - Liu, Hugo, 444
  - Lively elderly care system, 356
  - Lively safety watch, 358
  - local networking, 80–83
  - Lockitron connected door lock, 3
  - London
    - Metropolitan Police maps, 404
    - Oyster ecosystem, 141
    - Oyster travel card readers, 6, 43, 83, 137–138, 488, 618
  - “looks like” design prototypes, 580–586, 606
  - LOOP pelvic floor exerciser (case study), 413–418
  - Lowes Iris
    - home security motion sensor, 55
    - home system, 636
    - Safe/Secure DIY home security kit, 123

- Service Status screen, 486
- setup process, 473
- UI (plug switching), 65
- water heater, 373
- web UI device, 481–482

**M**

- Mace, Ronald L., 323
- machine-control of complex systems, 646–653
- macros and modes (programming), 637–643
- maintenance (physical devices), 267–272, 461
- maker culture, 600
- manufacturing
  - effects on environment, 459
  - physical devices, 267–272
- mapping connected products, 258–259
- Marble Answering Machine, 298–299
- Marks and Spencer department store, 42
- Mars Science Laboratory Curiosity Rover, 39
- mass market, innovation to.  
*See* innovation to mass market
- materials (physical device design), 267–272
- McDonough, William, 462, 463
- McEwen, Adrian, xix, 383, 505
- media from the future, 582, 586–588
- Mercedes-Benz dashboard app, 398
- messages, managing, 482–483
- Mesut, Jason, 256
- metadata, device, 657–663
- metaphors (product definitions), 195–198
- Metaphors We Live By, 196
- Metcalfe’s Law, 613
- microcontrollers, 36–37
- Microsoft Kinect, 295
- Miettinen, Markus, 512
- Mimo baby monitoring kit, 618–619
- Misfit Shine syncing process, 376–377
- missing data points, 41
- mobile devices
  - sensors in, 51
  - setting up, 507–508

- mobile interfaces for connected products, 315–317
- modeling data (prototyping), 597
- model making/prototyping, 251–252
- modes and macros (programming), 637–643
- Monk, Professor Andrew, 443
- monochrome displays, 288
- mood boards (design themes), 242–243
- Moore, Geoffrey, 116
- moral behavior (product values), 187–188
- +More platform (QR codes), 45
- Mother smart sensor system, 204
- motion detectors/sensors, 230, 49
- motivations/goals of users, 551–552
- MQTT protocol, 91–92
- Muji Multi Clock alarm clock, 300
- multimodal interactions, 312
- multiple devices
  - keeping track of, 617–619
  - multidevice services, 343
- multiple users of systems, 498–499
- multipurpose computers, 31
- Murphy, Niall, 98

**N**

- Nabaztag device, 34
- NAT: network address translation, 88
- NDEF (NFC Data Exchange Format), 44
- Negroponte, Nicholas, 646–647
- Nespresso Zenius coffee machine, 148
- Nest
  - Learning Thermostat, 6, 19, 33–34, 83, 112–113, 364–365, 560–561
  - Protect smoke alarm, 124–125, 226, 291, 295–296, 504
- Netatmo Weather Station, 40, 547
- Netflix, 350–351
- network communication patterns
  - IoT application protocols, 91–92
  - push/pull/polling, 87–90
- networking
  - intermittent connections for IoT devices, 59
  - Internet connectivity options, 77–80

- Internet networking, 76–77
  - Internet services. *See* Internet services
  - IoT architecture and. *See* IoT architecture
  - IoT UX relevance to, 58–59
  - IP communications to edge devices, 85–86
  - issues impacting UX, 64–65
  - latency/reliability impact on UX, 62–65
  - NAT: network address translation., 88
  - overview, 57–58
  - types of local, 80–83
  - unreliability of networks, 62
  - variations in latency/
    - responsiveness, 60–62
  - network interoperability, 388–393
  - Nexus Q (Google), 235
  - NFC (near field communication)
    - basics, 44–45
    - NFC-enabled smartphones, 7–8
    - overview, 84–85
  - Nielsen, Jakob, 60, 360
  - Nike Fuelband, 204
  - Nike+ Fuel Points, 393
  - Ninja Sphere gateway, 500
  - Ninja Sphere home lighting, 394–395
  - NodeRED (IBM), 642
  - nodes, sensor, 40
  - nonvolatile memory, 37
  - Norman, Don, 195, 218, 224–226, 341
  - notations, programming, 632
  - notifications, managing, 482–483
  - numeric comparison approach (device pairing), 515–517
  - Nyberg, Marcus, 594
- O**
- objectivity, data, 564–565
  - Object Push, 513
  - obstetric services experience mapping, 25
  - Octopus travel card, 6–7, 672
  - OECD privacy principles, 446–447
  - One Laptop Per Child security principles, 435–436
  - operating systems (OS), 38
  - operational context (research), 162–163, 169
  - Opower energy, 16–17, 185, 569
  - Oslo, Voy, 233
  - O’Sullivan, Dan, 276
  - out of band (OOB) pairing method, 517–518
  - Out of Box Experience Design, 477
  - output-only devices, 233
  - over-the-air (OTA) systems, 333
  - Owl Ugle (Voy Oslo), 233
  - Oyster travel card, 6, 43, 83, 137–138, 488, 618. *See also* Transport for London
- P**
- pairing devices
    - basics, 509–513
    - Bluetooth options for, 513–515
    - “just works” approach, 514–515
    - numeric comparison approach, 515–517
    - out of band (OOB) method, 517–518
    - passkey entry, 516–517
  - Palfrey, John, 385
  - Palmer, Chris, 425, 431, 433
  - paper prototyping, 602–604
  - pareidolia phenomenon, 265
  - Parker application (Streetline), 16
  - Parra, Jordi, 297–298
  - passively trackable objects, 42–48
  - passive sensors, 48
  - passkey entry (device pairing), 516–517
  - pattern libraries, 238
  - patterns of behavior (privacy), 445–446
  - Pay As You Drive program, 210
  - Payne, John, 192
  - Pebble
    - “just works” Bluetooth pairing, 515
    - Smart Watch, 35, 68–69, 378–379, 476, 479, 502
  - permissions (security), 434–435
  - personal assistants systems, 548–549
  - persona portraits (product values), 189
  - persuasion/behavior change (product definitions), 208

- pet ID microchips, 43, 509
- pet trackers, 50
- Philips Hue
  - bridge, 67
  - bulb, 89
  - system, 11
  - Tap light switch, 54
  - UI, 18
- photosensors, 48
- Physical Computing, 276
- physical controls, 278–282
- physical device design
  - aesthetics/appearance, 260–269
  - affordance, 258–259
  - anthropometrics/ergonomics, 257–259
  - BOM cost vs. future-proofing, 272–273
  - brand image/promise, 262–263
  - consistency/cohesion of products, 260–262
  - design freeze vs. continuous iteration, 238–240
  - form/function/usability, 257–260
  - form vs. function, 222–225
  - industrial design. *See* industrial design
  - levels of design, 226–228
  - materials/manufacturing/maintenance, 267–272
  - operating environments, 267
  - personality/character, 263–265
  - practicalities, 259
  - Product Designers, 237–238
  - sustainability/recycling, 271
  - user interaction/placement, 228–238
  - weight/texture/temperature, 266–268
- Physical Web project (Google), 46–47
- Pivot Power Genius, 258
- placement of physical devices, 228–238
- platforms
  - basics, 494–498
  - design of (IoT models), 26–27
  - following conventions for, 362–364
  - scaling (complex systems), 653–656
- plugs, controllable, 127
- point-to-point networks, 91–92
- portraits (product values), 189–191
- powering connected devices, 53–56
- powerline networking, 84
- preconfiguring system setups, 471
- principles of (product definitions), 199–204
- prioritizing product values, 190–191
- privacy. *See also* security
  - controlling information, 437–446
  - by design/by default, 448–450
  - legalities/consent/data protection, 445–448
  - overview, 436–437
  - “right to be forgotten”, 210
- product definitions
  - automation, 211–212
  - conceptual models, 195–198
  - data ownership, 209–211
  - definitions of value, 185–188
  - design principles, 199–204
  - domain metaphors, 195–198
  - generating the elevator pitch, 184–185
  - grounded innovation, 198–199
  - persuasion and behavior change, 208
  - questioning product strategies, 208
  - service ecology maps, 203–208
  - tools for discovering/communicating values, 188–195
- productization, 26, 112–115
- products
  - aesthetic aspect of, 135
  - alphabetical list of, 667–675
  - combining devices with services, 135
  - connected. *See* connected products research techniques
  - cost proportional to perceived value, 133–135
  - digital business models for physical, 147–149
  - interoperability with services, 399
  - maintenance/upgradeability of, 461
  - in new/existing markets, 121–123
  - product certification, 254–255
  - product designers, 237–238

- service, 115–116
  - solving real problems with, 128–133
  - versus tools, 123–128
  - types of data-driven, 544–549
  - profiles (product values), 189
  - programming
    - IoT devices, 11–12
    - user, 632–647
  - proprietary radio networks, 82–83
  - Proteus
    - connected pill (case study), 103–109
    - smart pill sensor, 40–41
  - prototyping
    - with collages, 582–584
    - with data, 596–599
    - deciding what to prototype, 603–606
    - feel of products, 590–594
    - overview, 577–582
    - product/service visualizations, 584–590
    - service design and, 594–596
    - user interfaces, 599–603
    - visualizing products in use, 582
  - Proverbial Wallets project, 303–304
  - proximity/visibility behavioral context, 165
  - public key encryption, 422
  - publish-subscribe model, 91
  - push-button configuration (WiFi), 505
  - push/pull/polling communication patterns, 87–90
- Q**
- QNX operating system, 38, 328
  - QR (quick response) codes, 42, 45–46, 310–311
  - quantified self products, 548–549
- R**
- radiation sensors, 55
  - Rawsthorn, Alice, 223
  - Reactable musical instrument, 300–301
  - read-only/read-write RFID tags, 42–43
  - real-time vs. historical data, 542–543
  - recycling (product manufacturing), 271
  - reflective design (physical devices), 218, 226
  - remote control, programming, 11–12
  - Renault Laguna car dashboard, 516–517
  - requirements (programming), 632
  - researching connected product design
    - actors and stakeholders, 154–162
    - behavioral context, 164, 169
    - context of interaction, 162
    - ecological context, 167–169, 169
    - expertise, 158
    - operational context, 162–163, 169
    - overview, 152–153
    - questions and concepts, 153–154
    - sociocultural context, 165–167, 169
    - techniques for. *See* connected products research techniques
  - ResearchKit (Apple), 550
  - ResMed sleep tracker, 549
  - responding to data (prototyping), 598
  - RESTful API, 399
  - Restriction of Hazardous Substances Directive (RoHs), 460
  - Revolv dashboard, 489
  - RFID (radio-frequency identification) tags
    - applications, 83
    - overview, 42–44
  - Rheingold, Howard, 651
  - robot vacuum cleaners, 534
  - Rogers, Everett, 116–117
  - Room-E project, 312–313
  - Rose, David, 305, 318
  - Rosen, Deena, 201
  - Rothensee, Matthias, 129
  - Rotich, Juliana, 608
  - routers, future, 235–236
  - Rowland, Claire, xv, 1, 29, 57, 111, 337, 381, 419, 467, 611
  - Royal National Institute of Blind People (RNIB), 321
  - RTLinux operating system, 38
  - rules (trigger-action programming), 637–638



## S

- Saffer, Dan, 348
- Sammy Screamer movement sensor, 81
- Samsung S Beam, 45, 84
- Samsung SCX-3405W laser printer, 506
- Samsung WW9000 connected washing machine, 142
- Sanders, Elizabeth, 170, 176
- San Francisco Exploratorium map, 194
- scaling platforms (complex systems), 653–656
- scaling the UX
  - addressing user needs, 621–623
  - beyond device-level controls, 620–621
  - capturing contextual information, 623–624
  - handling multiple devices, 617–619
  - overview, 617
- Schrage, Michael, 577
- Scout security system, 133
- Scratchable Devices project, 641
- screen interfaces
  - design fundamentals, 282–291
  - prototyping, 601–602
- Secure Simple Pairing (Bluetooth), 513
- security. *See also* home security systems; privacy
  - basics, 420–423
  - challenges of, 425–428
  - design requirements for, 429–435
  - impact on UX, 423–424
  - security alarm systems (services), 136
  - transparency/trust in UX design, 563–564
- Seedstudio, 600
- Segerstahl, Katarina, 340
- Sellen, Abigail, 448
- semantic consistency, 365
- Semantic Sensor Network Ontology (W3C), 658
- sensing data (prototyping), 597
- sensors
  - connected, 39–42
  - fundamentals, 48–53
  - sensor nodes, 40
- service design
  - overview, 24–25
  - prototyping and, 594–596
  - Service Design: From Insight to Implementation, 205
- services
  - building offerings of, 143–145
  - business models for, 145–146
  - combining with devices, 135
  - ecosystems of, 140–143
  - interoperability with products, 399
  - product/service visualizations, 584–590
  - service ecology maps, 203–208
  - service-led Internet setup, 501–502
  - service products, 115–116
  - service-to-service connections, 70–71
  - user understanding of, 136–140
- setting up systems
  - connecting devices, 478–479
  - context of use, 477
  - creating good experiences, 470–471
  - extra configurations, 478
  - flexibility/modularity, 471–474
  - instructions, 474–478
  - multiple installers, 478
  - overview, 469–470
  - preconfiguration vs. user effort, 471
  - testing setup, 479–480
- Seymourpowell consultancy, 243
- Shaping Things, xix
- sharing data (prototyping), 597
- shearing layers model, 133
- Shilton, Katie, 188
- Shneiderman, Ben, 11
- Shodan search engine, 428–429
- Sifteo Cubes, 299–300, 576
- SigFox, 79
- Siri Eyes-Free feature (SYNC3), 334
- Sketch-a-Move toy video, 589
- sketching (industrial design), 243–246
- Sketching User Experiences, 606
- Skydrop smart irrigation controller, 633
- Smart Cities project, 588
- smart drinks vending machines, 309

- smart meters, 80
  - smart parking applications, 16
  - smartphones
    - as gateways, 68–69
    - user interfaces, 491–492
  - smart plugs, 230, 631
  - smart sockets/outlets, 662
  - smart systems, 547
  - SmartThings
    - app, 484
    - Goodbye macro, 639
    - hub, 67
    - key fob, 481
    - product range, 263
    - Things list, 480
    - website, 390
  - social engineering, 450–458
  - sociocultural context (research), 165–167, 169
  - software
    - embedded, 37–39
    - engineering process, 633
  - Soundbox project, 505
  - spacial mapping (research), 171–172
  - Sparkfun, 600
  - specification, programming, 632–633
  - speech recognition, 291–292
  - Spotify
    - Android app, 363
    - Box, 297–298
    - iOS app, 363
  - Stajano, Frank, 424
  - stakeholders and actors, 154–162
  - Staples Connect
    - app, 408
    - hub, 396
  - Starck, Philippe, 226
  - states of being behavioral context, 165
  - static vs. dynamic data, 540
  - Stephens, Denise, 321
  - Sterling, Bruce, xix
  - Storehouse, 142
  - storyboards, 586
  - story collages, 584
  - SunSprite Tracklight, 343
  - SureFlap pet flap, 83
  - surface continuity (industrial design), 249
  - surge pricing (Uber), 543
  - Suri, Jane Fulton, 591
  - synchronization of data/content, 368–375
  - synergistic specificity, 356
  - systems. *See also* intelligent systems architecture
    - effects on UX, 71–72
    - interoperability levels of, 394
    - discovering presence of, 486–487
    - maintenance of, 485–486
    - system domain models, 663–664
    - system-generated metadata about devices, 658–661
    - system images, 342–343
- T**
- tactile interactions, 301–304
  - tactile quality of buttons, 280
  - Tado
    - homescreen, 495
    - thermostat/heating controller, 34–35, 313–314, 352
  - tangible benefits (product values), 187
  - tangible interactions, 297–302
  - Target “pregnancy coupons”, 186–187
  - task migration, 376–380
  - TCP/IP model, 76–77
  - technology layer (interoperability), 386
  - templates, design, 497–498
  - terminology, consistent, 361–362
  - Tesler, Larry, 348
  - Thalnic Labs, 413
  - themes, design, 242–243
  - Thingful search engine, 661
  - third-party data, 544–545
  - Thread standard, 86
  - Tiger, Lionel, 135
  - Tile Bluetooth tag, 231–232
  - Time It Right technology, 143–144
  - timelines (research), 172–174
  - time/place behavioral context, 165
  - timing/frequency of data readings, 543–544
  - Tony Tempa ear thermometer (BleepBleeps), 264
  - tools vs. products, 123–128
  - Touch device (3D Systems), 302–303
  - Toymail connected toys, 310
  - Toymail Mailmen, 266–267

Tractive pet trackers, 50  
Transmission Control Protocol (TCP), 76–77  
transparency/trust in UX design, 563–564  
Transport for London website, 619.  
*See also* Oyster card  
transport layer (networking), 76  
trigger-action programming, 636

## U

Uber car service, 160  
Uber surge pricing, 543  
UIs (User Interfaces)  
    generic vs. specific, 408–409  
    UI/visual design, 18, 21  
uncanny valley, 652  
under-mattress sensor, 443  
Union Pacific Railroad connected sensors, 529  
universal principles of design, 314, 322–324  
universal recycling symbol, 463  
upgradeability of devices, 461  
Usability Engineering, 60  
USA privacy laws, 445  
users. *See also* actors and stakeholders; consumers  
    behavioral changes, 565–568  
    controlling of permissions by, 434–435  
    control of complex systems by, 626–629  
    control via interoperability, 411  
    early/late majority, 117  
    home system user needs, 621–623  
    interaction with physical devices, 228–235  
    modeling for data, 665  
    motivations/goals of, 551–552  
    multiple, 498–499  
    programming by, 632–647  
    prototyping interfaces, 599–603  
    User Acceptance of the Intelligent Fridge, 129  
    user and design models, 341–343  
    user management, 483–485  
    user understanding of devices/services, 136–140

UX (User Experience)  
    business models affecting, 145–146  
    collaboration with industrial design, 255–257  
    creating good setups, 470–471  
    design, definition of, 3  
    Elements of User Experience diagrams, 19  
    evaluation of products, 607–608  
    for IoT. *See* IoT UX  
    latency/reliability impact on, 62–65  
    powering devices and, 55–56  
    productization and, 113–114  
    risks of autonomous systems, 650–653  
    scaling. *See* scaling the UX  
    scaling up in platforms, 653–656  
    impact of security on, 423–424  
    system architecture effects on, 71–72  
    UX/Industrial Design  
        collaboration, 256

## V

value propositions (products)  
    clearly communicated, 128  
    convincing users of, 120–123  
    definition of, 112  
    for IoT, 120–123  
    UX driven by, 114  
values, product  
    defining, 185–188  
    definitions of, 185–188  
    prioritization of, 190–191  
    tools for discovering/  
        communicating, 188–195  
vending machines in Japan, 452–454  
verbal explanations (research), 176–177  
vibrobelt GPS device, 302  
Victor, Bret, 275  
video prototyping, 588–590  
visceral design (physical devices), 218, 226  
visualizing data, 569–571  
visual/screen interfaces, 282–291  
Vitality server, 204–205  
voice interfaces, 290–294  
VxWorks operating system, 38–39

## W

Walton, Philip, 163  
WarKittah cat collar, 425  
Waste Electrical and Electronic Equipment Directive (WEEE), 462  
Waze navigation app, 535  
weather apps, 556–557  
weather forecasting device (Ambient Devices), 306  
web APIs, 95  
web interfaces for connected products, 315–317  
websites for downloading  
  Designing with context, 305  
  RNIB checklist, 321  
websites, list of company, 667–675  
WEEE (Waste Electrical and Electronic Equipment)  
  symbol, 254  
Weightless, 79  
Weiser, Mark, 558  
WeMo. *see* Belkin  
Wendel, Stephen, 565  
Whirlpool laundry appliances, 38  
Whyline programming tool, 643  
WiFi  
  OOB pairing for, 506  
  overview, 78–79  
  setting up devices for, 504–505  
  WPS (WiFi Protected Setup), 505  
WigWag connected home kit, 636  
Wii Remote/WiiMotionPlus, 51  
Williams, Luke, 246  
Wilson, Mark, 142, 250

Wilton, Denise, 112  
Windows Phone 8, 626–627  
Wink gateway, 391  
Wink hub, 507  
Withings  
  ecosystem of devices, 15, 350  
  iPhone app, 51  
  mobile app, 355  
  Smart Body Analyzer bathroom scale, 33, 318–319  
Wizard-of-Oz prototyping, 592–594  
“works like” prototypes, 606  
WPS (WiFi Protected Setup), 505

## X

X10 Powerhouse, 119  
Xbox online multiplayer, 293  
Xively public API, 397, 400–401

## Y

YouGov profiling service, 444  
Yu, Suelyn, 201

## Z

ZigBee  
  pairing devices via, 511  
  protocol, 68–69, 72  
  radio network, 82  
Zipcars, 459, 8  
Zite.com, 451  
ZWave  
  pairing devices via, 511  
  radio network, 82

## About the Authors

**Claire Rowland** is an independent UX and product strategy consultant based in London. She has a particular interest in taking connected products from an early adopter user base to the mass market, and technologies that support mundane, everyday activities.

Before becoming independent, she worked on energy management and home automation products as the service design manager for *AlertMe.com*, a connected home platform provider. Previously, she was head of research for design consultancy Fjord, where she led EU funded R&D work investigating the interusability of interconnected embedded devices. She has worked in UX design and research for mobile, multi-platform, and web services since 1997. She tweets as @clurr.

**Elizabeth Goodman** investigates the possibilities and perils of emerging technologies in her design, research, and writing. Her 2014 PhD from UC Berkeley's School of Information focused on human-computer interaction and design practice for novel technologies, and was supported by National Science Foundation and Intel fellowships. In 2013, she authored the second edition of *Observing the User Experience*, a widely used handbook of design research methods. Her work has taken her from exploratory research and design teams at Intel, Yahoo!, and Fuji-Xerox to her current position with 18F, a service and product design group within the US government aimed at making federal agencies more efficient, more transparent, more accessible, and more accountable to the people they serve. Elizabeth speaks widely on the design of mobile and pervasive computing systems at conferences, schools, and businesses. She lives in San Francisco with her husband and two children. She tweets as @egoodman.

**Martin Charlier** is an independent design consultant based in London and a cofounder of Rain Cloud. Martin is a strategic designer with experience across new media art, industrial design, interaction design, and service design. His speculative design work has been awarded by the Royal Society of Arts, and he is a fellow of the organization.

He has previously worked at innovation firm Frog Design, cutting-edge art collective Random International, and digital service design consultancy Fjord. He tweets as @marcharlier.

**Ann Light** is Professor of Creative Technology at the University of Sussex, with a background in qualitative research exploring the impact of current and future design choices. Her fascination with digital mediation led to studying websites and online discussion as early as 1995, and by 2002 she was commenting on the challenges of using networked products. A focus on connected and ubiquitous contexts of use has included looking at design globally—with projects in Ghana, India, Chile, and Uganda, workshops on six continents, and a role advising the European Union on the future of the Internet. She works on design for social change and the politics of participation using creative methods across much of her research. She also publishes on social innovation, human-computer interaction, and cross-cultural methodology.

**Alfred Lui** has been a user experience designer for consumer products since 2004. He created user interfaces and digital services for companies around the world, including The BBC, Motorola, PayPal, Jawbone, and SmartThings. In London, he was part of a EU-funded research project to investigate best practices in designing networked objects. Soon after he moved to San Francisco in 2011, he founded the San Francisco Internet of Things Meetup to build a local community around the topic. Alfred frequently writes and speaks about user experience design for the Internet of Things. He tweets as @connection76.