O'REILLY®

User Experience Design for the Internet of Things

Why It's More Than UI and Industrial Design

Claire Rowland
with Martin Charlier

Short. Smart. Seriously useful.

Free ebooks and reports from O'Reilly at **oreil.ly/fr-design**

Free ebooks, reports and other articles on UX design, data-informed design, and design for the IoT. Get insights from industry experts and stay current with the latest developments from O'Reilly.

O'REILLY®

User Experience Design for the Internet of Things

Claire Rowland

With Martin Charlier

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

User Experience Design for the Internet of Things
by Claire Rowland

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 (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Angela Rufino
Acquisition Editor: Mary Treseler
Production Editor: Colleen Lobner
Copyeditor: Jasmine Kwityn

Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Rebecca Demarest

September 2015: First Edition

Revision History for the First Edition
2015-09-14: First Release

978-1-491-93393-0

[LSI]

Table of Contents

CHAPTER 1

User Experience Design for the Internet of Things

Introduction

When we think of design for connected products, we tend to focus on the most visible and tangible elements. These are the industrial design of connected devices, and the user interfaces (UIs) found in mobile and web apps and on the devices themselves. They are important concerns, which have a major impact on the end user's experience of the product. But they're only part of the picture. You could create a beautiful UI, and a stunning piece of hardware, and users could still have a poor experience of the product as a whole.

Designing for the IoT is inherently more complex than 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 compelling consumer IoT value propositions. Some of this stems from the fact that there are more aspects of design to consider. Tackling them independently creates an incoherent user experience (UX).

Designing a great connected product requires a holistic approach to user experience. It spans many layers of design, not all of them immediately visible. More than ever, it requires cross-discipline collaboration between design, technology, and business. Great UX may start with understanding users. But the designer's ability to meet those users' needs depends on the technology enablers, business models, and wider service ecosystem.

As designers and their collaborators, we need a shared understanding of the challenges. We also need a common vocabulary for discussing them so that when we use the word "design," we're talking about the same things.

This report introduces a framework for understanding the experience design of consumer IoT products. It sets out the different facets of design that combine to shape a connected product, and shows you how they fit together. It explains the extra complexities that surround designing for connected products. Finally, it discusses how technology and the wider commercial context work to shape the UX of IoT products.

It's beyond the scope of this report to delve into the design process for the IoT. This is more complex than pure software design: hardware adds new considerations, and is less easily modified. Value propositions and design requirements must be clearly defined before product and design decisions are baked into the hardware, when they are hard to change. But here, we will show why the nature of the challenges requires collaboration between design and engineering for both hardware and software, and the business.

NOTE

In this report, we use the term "product" to refer to any offering that solves a problem for people or fulfills a need in daily life. That could be a physical device, a web or other service, or most often, a combination of those. We focus on designing for consumer products, as that is the authors' expertise. But many of the principles and design challenges described will also have analogs in commercial or industrial settings.

Why UX for IoT Is Different

Connected products pose design challenges that will be new to designers accustomed to pure software services. Many of these challenges stem from:

- The specialized nature of IoT devices
- Their ability to bridge the digital and physical worlds
- The fact that many IoT products are distributed systems of multiple devices
- The quirks of networking

How tricky those challenges prove will depend on:

- The maturity of the technology you're working with
- The context of use, and the expectations your users have of the system
- The complexity of your service (e.g., the number of devices the user must interact with to use it)

But for most connected products, you'll need to consider the factors described in the following subsections.

Specialized Devices, with Different Capabilities

Many of the "things" in the Internet of Things are specialized, embedded computing devices. Unlike general-purpose computers (smartphones and PCs), their hardware and software is optimized to fulfill specific functions.

Their physical forms must be designed and engineered. Their UI capabilities may extend from screens and buttons into physical controls, audio, haptics, gestures (see Figure 1-1), tangible interactions, and more. But user interactions must be designed without the benefit of the style guides and standards that web and mobile designers can rely upon. Some may have no user input or output capabilities at all. The only way to find out what they are doing or what state they are in may be via a remote UI.

Real-World Context

Connected products exist in the physical world. Sensors enable us to capture data we did not have before for digital transmission, allowing us to take more informed actions in the real world. Actuators provide the capability for digital commands to produce real-world effects (see Figure 1-2). They can be remotely controlled, or automated. But unlike digital commands, real-world actions often cannot be undone.

Figure 1-1. Jared Ficklin demonstrates Room-E, a system combining voice and gestural interactions; a gesture (pointing at a lamp) is combined with voice input ("computer, turn off this light") to turn off the correct lamp[1] (image: Taylor Hamilton, frog, and Jared Ficklin).

The physical context of use creates further challenges. Devices may need to be ruggedized for outdoor use. An in-car system needs to be designed to minimize distraction while driving. A remotely controlled oven needs to minimize the risk of fire. Devices must adhere to regulatory requirements such as radio interference or waste recycling standards. And the social context of use may be particularly complex, especially in the home. Techno-centric solutions which are insensitive to the needs of the occupants will fail. For example, an assisted living product needs to balance the need of vulnerable people for safety and support, while preserving their privacy and autonomy. Automated rules and modes in some smart home systems perform actions when certain conditions are met, like turning devices on or off when people arrive home, wake up, or leave. And permissions in some smart home systems allow an "admin" user to grant or deny access to certain devices to others in the house, such as controlling TV or game console time for children or locking cupboards containing dangerous things. But both of these often fail to take into account that real life, especially in families, is often messy and unpredictable. It's not always possible to predict which devices

1 Jared Ficklin is a frog fellow and the Chief Creative Technologist at argodesign. Find him on Twitter @jaredrawk or at *argodesign.com*. Watch the Room-E demo on YouTube.

will be needed or not needed at different times. And in most families, permissions are often flexible and negotiated. Few people enjoy feeling like sysadmins for their own homes.

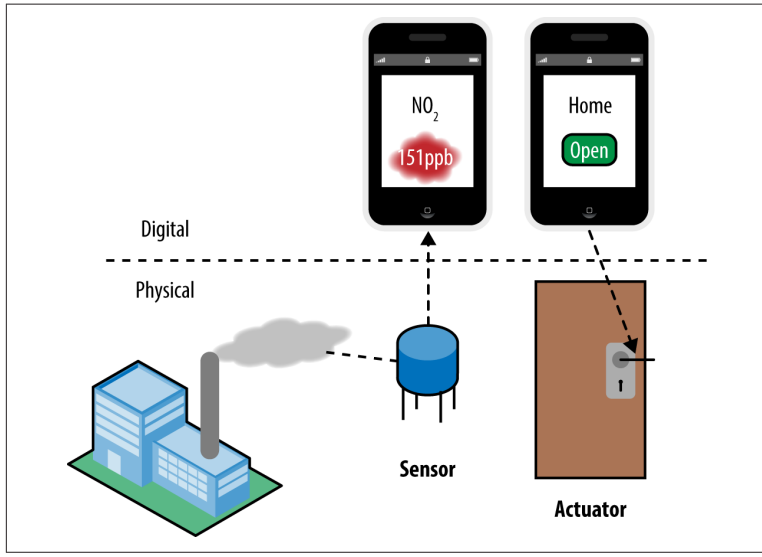

Figure 1-2. Sensors convert readings from the physical environment into digital information; actuators convert digital instructions into mechanical actions.

Designing for Systems of Devices and Services

Many connected products are systems of diverse devices and services. Functionality may be distributed across multiple devices with different capabilities.

Designers need to consider how best to distribute functionality across devices. They need to design UIs and interactions across the system as a whole—not treating devices as standalone UIs—to ensure that the overall UX is coherent. This is *interusability*. And much of the information processing for an IoT product will often happen in the Internet service. So the whole system experience is often equally or more important than any single device UX.

Furthermore, they need some understanding of how the system works. Even quite simple connected products are conceptually more complex than non-connected ones. Code can run in more places. Parts of the system will inevitably go offline from time to time. When this happens, basic knowledge of which component does

what will help users understand the consequences, and figure out what action may be required.

Many connected products support automation—for example, home automation rules that turn devices on and off in response to pre-set triggers. Users may need to keep track of a web of interrelationships among devices to predict, understand, and fix undesired clashes and strange behaviors.

Over the last 30 years, the prevailing trend in UI design has been direct manipulation.[2] Users control visual representations of objects and immediately see the outcome of their actions, which can be reversed (see Figure 1-3). But many IoT interactions are displaced in location (remote control) or time (automation). This breaks the link between user actions and visible, reversible consequences we have come to expect from modern software.[3]

Complex products, like a connected home system, can have many users, multiple UIs, many devices, many rules and applications. Understanding and managing how they all interrelate can be extremely difficult.

Aside from the configuration overhead this imposes on users, this is a cognitive challenge. Most of us are good at thinking about concrete things. But when it comes to understanding systems and interrelationships, and predicting the future consequences of our actions, we often struggle.

Designing for Networks

Another major factor is the impact of the network on UX. Designers from web and mobile software backgrounds have the luxury of assuming that devices will be nearly always connected. And most users understand that sometimes the Internet, as experienced through PCs or mobiles, can be sluggish or unreliable. Emails can be slow to download and Skype calls can fail. When latency and reliability problems do occur, they may be frustrating but are not unexpected, and can be worked around.

---

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

3 HCI researcher Alan Blackwell, in conversation. See Alan F. Blackwell, "What Is Programming?" *Proceedings of PPIG* 2002: 204–218.

Figure 1-3. An image by the designer Susan Kare, created in MacPaint 1.0: an early example of a popular direct manipulation interface (image: Wikicommons).

Our experience of the physical world is that things respond to us immediately and reliably. Light switches do not "lose" our instructions or take 30 seconds to produce an effect. Delays and glitches are inherent properties of physical networks and transmission protocols. But they may feel strange experienced through "real-world" things. It's impossible to engineer these issues entirely out of any Internet-connected system (see Figure 1-4).

In addition, the nature of connected devices is that they often connect only intermittently, in order to conserve power. Computers promise to provide us with precise, accurate, and timely data about the world around us. But distributed IoT systems may not always be in sync, and different devices may therefore report different information about the state of the system.

In a distributed system, designers must often consider delays, discontinuities, and uncertainty as part of normal user interactions and handle them as elegantly as possible.

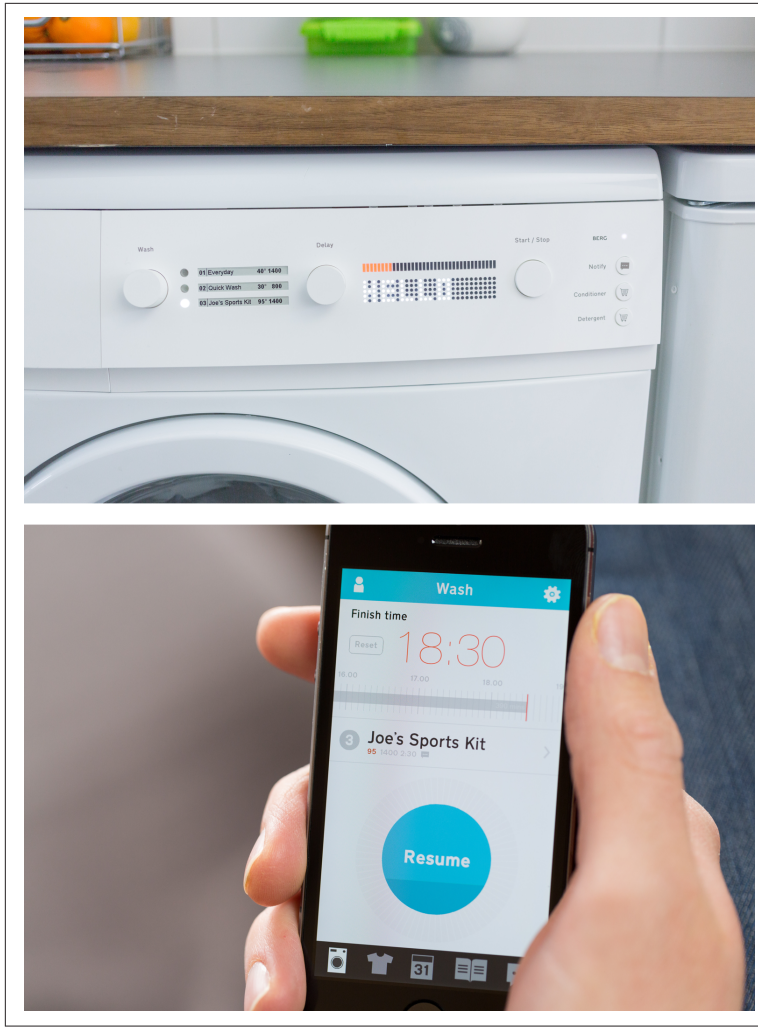

Figure 1-4. The design and product company BERG produced a beautiful concept prototype for a connected washing machine. The video shows instant responses between the mobile app and washing machine, running over the Internet. In a real-world situation, this could not be guaranteed (images: Timo Arnall of BERG).

A Framework for IoT Design

The most visible and tangible design elements of an IoT product are:

- The user interfaces/visual and aesthetic design of the devices: whether web and mobile apps, or on the connected devices themselves
- The industrial design of the physical hardware: the form factor, styling, and capabilities of the connected devices

UI and industrial design are important, but they are not the whole picture. It's possible to create apps and industrial design which individually seem appealing, but for the overall UX still to be poor. This can happen if the components don't work well in concert, or the value proposition is not a good fit for the user's motivations for using the product. It can also reflect limitations of the technology or service ecosystem, which prevent the product from fulfilling surrounding user needs as well as it could.

The UX is not just shaped by what the user can see or encounter directly. The basis for a valuable, appealing, usable, and coherent IoT product is created by care for the UX at less visible, system-oriented and strategic levels. This requires a good underlying technical, service, and product framework aligned around user needs. It requires attention to the experience of using the system as a whole.[4]

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.

4 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. 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. The model highlighted dependencies between work that was not directly apparent to the user but which determined aspects that they could directly experience. In designing for connected products, there is a similar pattern, in which strategic and technical design is needed to enable good UX at the UI and device level.

The facets of design that must be integrated to deliver a good UX for a connected product are set out in Figure 1-5.

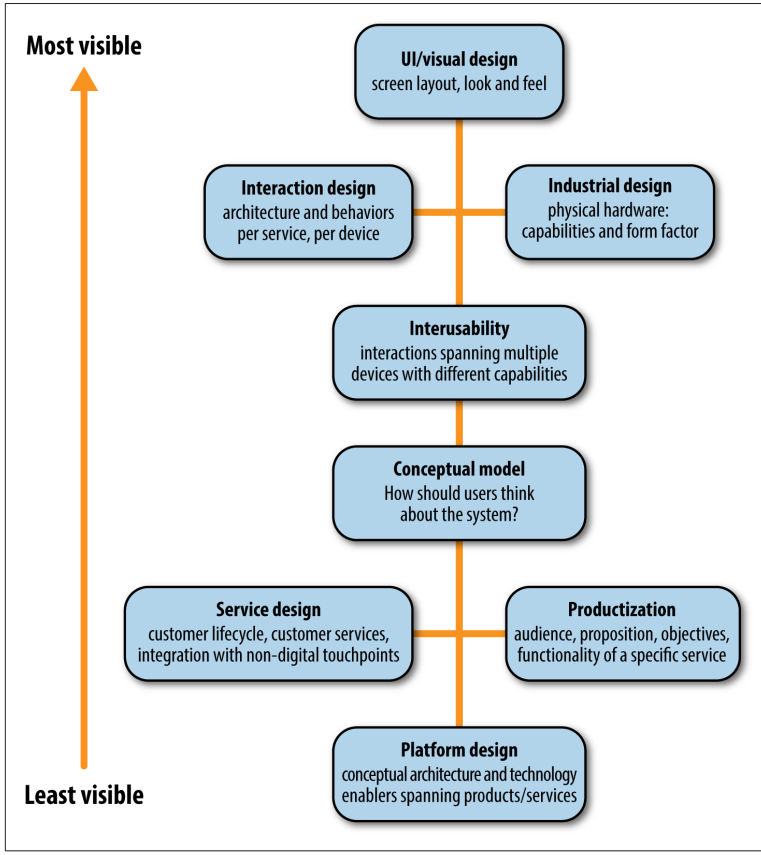

Figure 1-5. Facets of design in IoT: a good product requires integrated thinking across all of these.

This is not a set of activities or job roles. It says nothing of user research, or data science, and links to engineering are represented only at the platform level. It's a design-centric view of the aspects of design that need to be considered and integrated.

Some of these, like productization and platform design (e.g., APIs and data models), we may not conventionally think of as the realm of UX design at all. They may well not be the designers' responsibility. But they are all enablers (or blockers) to the end user experience. Those who are responsible for them must understand the interdependencies between their work and collaborate.

Some of these facets are separate concerns but will evolve in tandem. For example, UI, interaction design, and interusability need to be thought about together.

Depending on the type and complexity of your service, layers will require more or less of your time. UX thinking at the platform layer will initially be a case of understanding relatively simple APIs and data. It will become considerably more complex—and require more intense collaboration between design and engineering—once you start adding multiple devices to a service. But if your system grows and becomes more complex over time, the parts that are less important to you now may become more so in future. It's worth being aware of the whole picture, even if not all of it is relevant to you right now.

UI/Visual and Interaction Design

UI design is the most concrete level of UX. This is the form that a device interface takes. On a screen interface, it refers to screen layout, the choice of controls and UI elements, visual styling, and look and feel.

A connected product may have multiple UIs, from web and mobile apps, to specialized embedded device interfaces. Devices may have novel and advanced interfaces that move beyond screens and buttons to audio, voice, haptics, tangible interactions, gesture, computer vision and biometrics (from heart rate to brain waves). For a gestural or speech interface, key UI concerns might be defining the precise gestures for particular commands, or the tone of voice, phrasing, and vocabulary used by the voice system.

In the diagram, we differentiate between UI design and interaction design.

Interaction design is the design of system behaviors: it considers how the UI is used over time. Interaction designers shape the sequences of actions between the user and the device needed to achieve particular goals or activities (see Figure 1-6). 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.

Figure 1-6. Device and app interaction design concepts for Hackaball, a programmable ball for children (images: Map and Made by Many).

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. Interaction design is primarily concerned with behaviors and actions. UI design (particularly visual UI design) is concerned with layout and styling. In visual interfaces, they are easily separated, although for less conventional interfaces, like tangible or audio interfaces, UI and interactions may be much more closely intertwined. Hence in this report, we highlight that they can be separate concerns but discuss them together.

New Opportunities in UI and Interaction Design

A mobile and/or web app is a common component of most connected products. There are many good resources on designing for web and mobile. So in this report, we've focused on the opportunities to use less conventional interface types on embedded devices.

Connected products create new opportunities for interacting with digital devices. Touchscreens have their place and are adaptable to the needs of general-purpose computing platforms. But in the physical world, we use our hands and senses in many different ways on a daily basis. Specialized devices can enable designers to explore a

wider range of user input and output methods that are better suited to the function of the system (see Table 1-1).

Table 1-1. Alternative input and output channels

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

In the following subsections, we briefly discuss some of the interface and interaction options that can be employed on specialized devices.

Physical Controls

Physical controls are all around us: push buttons, switches, and sliders and rotary knobs that let us set a value on a range or select between multiple settings. They aren't just input methods, they sometimes display state—for example, the dial on a washing machine doesn't just allow you to choose a program, it also shows you which program is running.

Physical controls can be more satisfying to use than touchscreens. They tend to be more accessible to users with vision impairments, who memorize button positions or affix tactile labels. But a key benefit of a connected product is the ability to modify or upgrade the device over time. Baking functionality into fixed physical controls can make it harder to do this.

A further complication is the need for the physical control to communicate state accurately, when settings can be changed from

remote devices too. For example, conventional dimmer switches use a dial with minimum and maximum settings. If the light is dimmed from a smartphone app, what happens to the physical switch? If it's set to maximum but the bulb is dimmed from the app, the switch no longer reflects the state of the bulb. It cannot be used to turn up the brightness as it's already at the maximum point. The physical switch could be motorized, turning in response to digital commands. Or it could be redesigned so that it can display the correct status and will not get "stuck" on a minimum or maximum setting (see Figure 1-7).

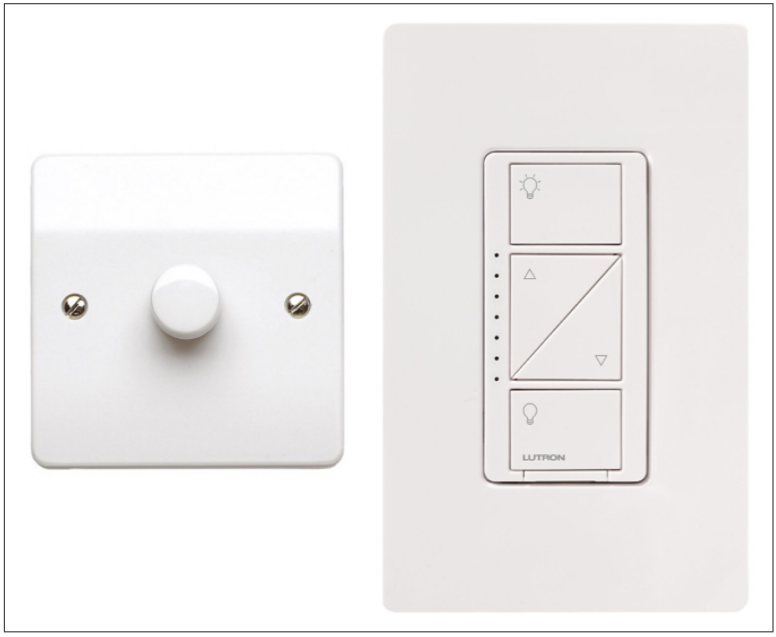

Figure 1-7. A conventional dimmer switch, and a connected dimmer; note the LED strip down the left, used to indicate dimming level (image: Lutron).

Visual and Screen Interfaces

Most software designers are accustomed to designing for high-res color screens. But there are a range of simpler display types that are often used on connected devices.

Almost every electronic device has at least one LED to indicate that it is switched on or to communicate some kind of status. Connected devices have a wider range of potential statuses to communicate (e.g., booting, waiting for activation/WiFi network info, online

or offline). Light can even be the main interface, such as the Ambient Orb.

Basic LCD screens may be able to display alphanumeric characters, or fixed custom graphical segments. In fixed segment displays, every possible piece of information needs to be designed into the display upfront (Figure 1-8). So a heating controller that can show temperature in Celsius and Fahrenheit must have both C and F characters. Pixel matrix displays are more flexible. (Smartphones and laptops use high-end color matrix displays). E-ink displays, like Kindle, use less power and can show something when a device is off. But they have a low refresh rate so aren't suited to animations, like scrolling.

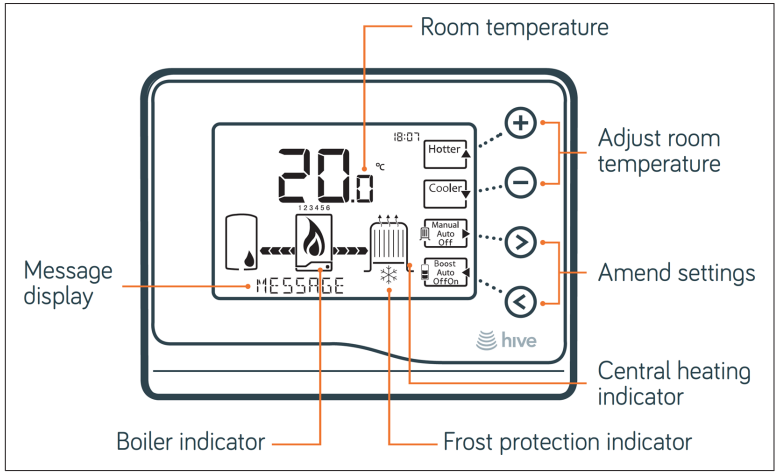

Figure 1-8. Design for a thermostat LCD display combining fixed segments and a character set; the message display is dynamic, but all the graphical elements need to be designed into the screen prior to manufacturing—there is no option to change them later (image: British Gas).

Screens can communicate a lot of information, but they increase the cost of device, and can make it harder to eliminate feature creep (i.e., to keep the device simple and focused).

Using visual channels for system input, through computer vision and recognizing QR codes, is also an option. But they can sometimes be a clunky or unreliable option.

Audio and Voice

Audio interfaces allow devices to communicate when the user is not looking at them. Even simple sounds have a stronger emotional character than simple visual interfaces, from urgent high-pitched beeps to the satisfying swoosh of an email sending. Care must be taken not to irritate or overwhelm the user, or make noise at inappropriate times (e.g., when someone may be sleeping).

Voice interfaces are a powerful way to input or output fairly complex information, and users can operate them while doing other things. However, they are linear—the system can only present the user with one option at a time. For efficient input, the user needs to remember which kinds of command the system can understand. The tone and accent of the computer's voice can be very loaded with meaning: for example, what sounds friendly (or competent) in one culture may sound cheesy (or aggressive) in another. At the moment, they are also often unreliable, may support a limited range of languages, and often require an Internet connection for server-side processing.

Gestural Interfaces

Swiping and tapping on touchscreens are forms of gestural input. But using computer vision, devices like the Kinect can also recognize mid-air gestures. Connected home systems such as Ninja Sphere and Onecue have experimented with gesture controls for lighting and home hubs.

Gestural inputs work well for gaming, and short interactions where the commands are obvious. Longer interactions can lead to fatigue and muscle pain, and false positive inputs can be an issue (see Figure 1-9).

Tangible and Tactile Interactions

Tangible user interfaces give physical forms (such as tokens) to digital information. They enable direct manipulation through physical interaction. They can be great for interactive experiences in museums, educational products, or for musical instruments (see Figure 1-10). However, losing tokens or parts of the interface can render them unusable, and thus far, there are few commercial products that use tangible interfaces.

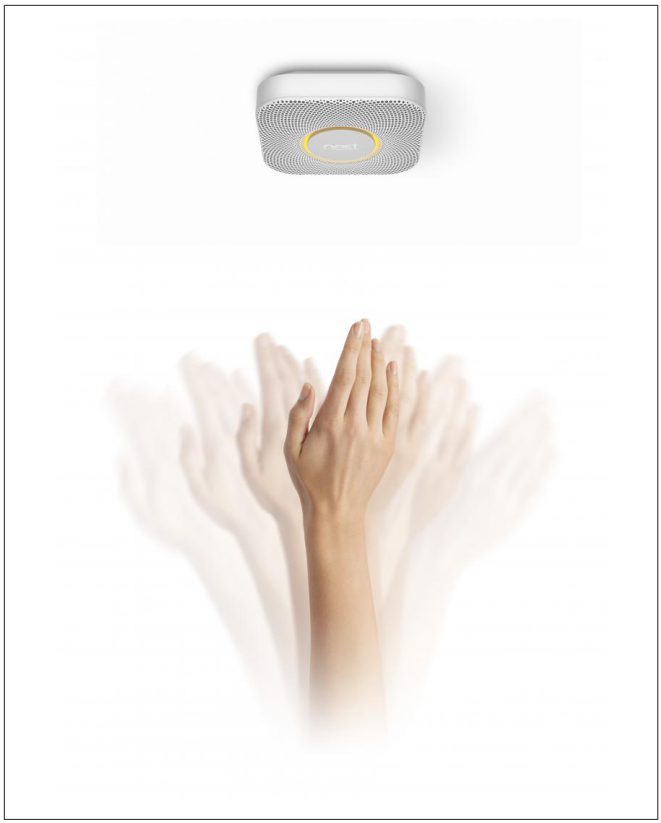

Figure 1-9. Nest's wave-to-hush feature allowed users to silence the smoke alarm; it was deactivated when it was realized that users might wave their arms in panic during a real fire and unintentionally deactivate the alarm (image: Nest).

Haptic output uses tactile actions to convey information, such as vibration or taps. It can be subtle, and demands less attention than sound or visual UIs. Vibration output aside, haptic interactions are currently mostly seen in research projects. The Apple Watch Taptic Engine (which uses an actuator in the watch to "tap" the user on the wrist) is one of the first mainstream commercial products to use haptics; see Figure 1-11.

Context-Sensitive Interfaces

Connected devices may create new possibilities for us to interact with the world, but user attention is finite. *Context-sensitive inter-*

faces can reduce complexity by tailoring interfaces based on context. They sense data points that are proxies for context, such as time of day, weather, location, movement, and the identity of the user. This is used to make inferences about the user's needs, presenting only the most relevant options or even taking autonomous action.

Figure 1-10. The Reactable is a musical instrument with a tangible interface (image: Reactable/Massimo Boldrin).

If this is executed in a genuine and smart manner, it can be a powerful way to reduce information overload. When executed poorly, it can appear patronizing or overbearing.

The idea that we could dispense with explicit UIs altogether ("no UI") in truly "smart" environments that anticipate and adapt to our needs is a seductive one. But in practice, inferring user needs and the context of use and how to respond appropriately is often a hugely complex task. If a person walks into a room at night, when should lights be turned on, and which ones? Do they need dim lighting for a nighttime bathroom trip, or bright lighting for an activity like reading? Is someone else trying to sleep? In that case, dim or no lighting might be required to avoid waking them. But if the person entering is an intruder, waking them up is crucial. Smart systems will often make incorrect inferences. Users need visibility of what the system is doing, and why, so that they can understand and (if necessary) correct any problems.

Figure 1-11. The Taptic Engine is a linear actuator inside the Apple Watch that can tap the user on the wrist (image: Apple).

Accessibility

Abled designers often forget that the products they work on can be much harder, or impossible, for people with disabilities to use. Sensory, motor, and cognitive disabilities can all affect the user's ability to perceive what the device is doing, and control it. It's worth remembering that all of us are, in practical (though not social) terms, sometimes "disabled." In a noisy environment, none of us can

hear well, and though we may have two hands, if one is taken up holding a child or a heavy bag, we only have one left to interact with a product.

Connectivity opens up an important opportunity to create more accessible products. It allows users a choice of interaction channels. Smartphones and the Web can provide more accessible UIs for devices that, on their own, are not accessible. For example, a thermostat UI may be inaccessible to someone with a severe vision impairment. But an accompanying iOS app designed to work well with Apple's VoiceOver opens up access to a wider range of users. The haptic interface on the Apple Watch can provide a communication mechanism for those with vision and hearing impairments. Users could, in theory, bring their own interfaces, already adapted to their abilities, to control any device around them. Personal, connected devices can also be used to identify the user's needs to adaptive environments (see Figure 1-12).

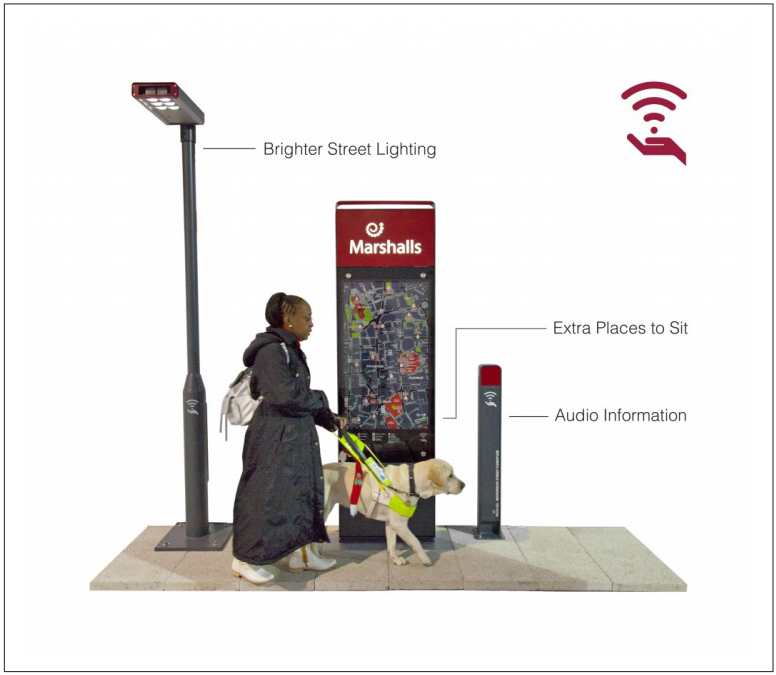

Figure 1-12. Adaptive street furniture design, using a smartphone or keyfob ID to identify the user's additional needs to trigger adaptations in lighting, seating, and audio descriptions of the environment (image: Ross Atkin Associates for Marshalls).

Industrial Design

Designing great connected products requires deep collaboration between hardware and software designers. Industrial design refers to the aesthetic and functional design of the physical hardware in the service, covering the choice of form, materials, and capabilities it may have.

Designing hardware 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 stems from a different mindset and approach to UX. It is a less task-driven approach—it involves equal parts thinking about aesthetics, usability, engineering, and manufacturing. 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.

The aesthetics and appearance of a device play an important role in conveying brand values through a common design language for a company's products (see Figure 1-13) and in making products desirable and enjoyable. They can also improve the usability of a product.

Figure 1-13. Korea Telecom range of products unified by a common design language (image: Seymourpowell).

Basic considerations for the physical design include:

- How frequently the user will interact with it
- How conspicuous it will be once installed
- Whether users will own multiples of it

Traditionally, the physical form of a device would be expected to represent its function. But there are few established archetypes or conventions for connected objects. And many functions may not have an obvious physical form. What does a network gateway look like?

But physical appearance is only the tip of the iceberg. Physical requirements also influence the design of a product. The need to withstand environmental conditions such as heat or rain, and material degradation, must be considered. There must be space to accommodate electronic circuitry. If there is a radio antenna, it must get a clear signal. This imposes particular requirements on material choice (metal casings may be an issue) and component placement. Manufacturing constraints limit the forms that can be made. In addition, devices must be certified for compliance with radio interference and safety regulations.

Most importantly, as consumption moves back from digital to physical, all designers involved need to act responsibly. Software apps are easily deleted when they fall from favor. But physical products don't disappear. When we throw them out, they end up in piles of e-waste, often in poorer nations. They are burned and dismantled to salvage the components and metals that can still be sold, and in the process, toxic materials not only contaminate the environment but also poison the workers that scrap them. Connectivity should enable us to upgrade and improve products through software, extending their lifespan. But too often it results in products becoming obsolete faster as our expectations rise or the companies that made them go out of business.

Interusability

Conventional usability/UX is concerned with interactions between a user and a single UI. But connected products are systems of devices and web services. There are often multiple devices through which the user interacts with the system. Designers can no longer consider the UX of a single device UI in isolation. The UX needs to feel coherent across the system as a whole, even when the devices involved may have quite different form factors and input/output capabilities. This is *interusability*.[5]

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.

Composition

In a distributed system, designers need to decide which device should do what, in terms of user-facing functions. Will each device have a specialized, unique role, or will some functionality be available across more than one device? This is *composition*. Appropriate composition takes into account the capabilities of each device and the context of use. One typical decision in consumer IoT is whether to bake functionality into a hardware interface, or offload it to a mobile or web app (see Figure 1-14). The latter keeps hardware costs down, and means it's easier to update and redesign the UI to support new functionality. But it carries the risk that the user must always have connectivity, or a working phone/PC, to use the device. And sometimes users need, or prefer, to have on-device controls.

5 This model was originally proposed in Minna Wäljas, Katarina Segerståhl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen's paper "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 '10*.

Figure 1-14. All interactions with the Bluespray irrigation controller are handled via a smartphone app, while the skydrop has full controls on both device and smartphone app (images: Bluespray and skydrop).

Consistency

Designers also need to determine the appropriate degree of consistency across different system UIs and interactions. Which elements of the design—such as terminology, platform conventions, aesthetic styling, and interaction architecture—should be the same? And which should be different?

Jakob Nielsen's classic usability heuristics state that "Users should not have to worry whether different words, situations or actions

mean the same thing. Follow platform conventions." But where devices have different form factors, the idea that design elements should appear the same can be in tension with the need to follow very different platform conventions.

In the author's experience, the top priority is terminology. Identical functions must have the same name across all devices. The second priority is following platform conventions—or at least being "true" to the device. Established design guidelines exist for mobile and some wearable platforms, like iOS/Apple Watch and Android/Android Wear. A good mobile app should be a good mobile app, not one that attempts to mirror hardware interactions on a touchscreen. The Nest thermostat and app are a good example of appropriate cross-platform consistency in aesthetics and interactions (see Figure 1-15).

It's less important that interaction architecture—the logical structure of UI features and controls—be the same. Different features may be prioritized on different devices. Devices with limited UIs may need deeper functional hierarchies. They may be optimized for a few key features, at the expense of others, which are possible but hard to find or inefficient to use. Designers may also need to work with less than ideal (e.g., heavily modal) UIs on legacy or low-cost hardware. These should not unnecessarily constrain the design of other interfaces that are under the designer's control.

Continuity

Continuity is the flow of interactions and data in a coherent sequence across devices. It creates the sense for the user that they are interacting with the service, not with a bunch of separate devices.

This does not necessarily mean designing seamless interactions. It often means handling interstitial states gracefully: designing for the spaces between devices. Network latency and reliability issues mean that designers may have to handle delays and failures in the interface as part of "normal" use. There could be a delay of seconds or even minutes between pressing a button to turn on a light, and the light actually coming on. In a conventional UI, the same change of button state can be used both to acknowledge the user's command, and to confirm that it has been executed. But in a distributed system, these may need to be represented separately. Acknowledgement of the user's command needs to be shown immediately, but there may be a

delay before it's possible to confirm that the target device has actually acted on the command (see Figure 1-16).

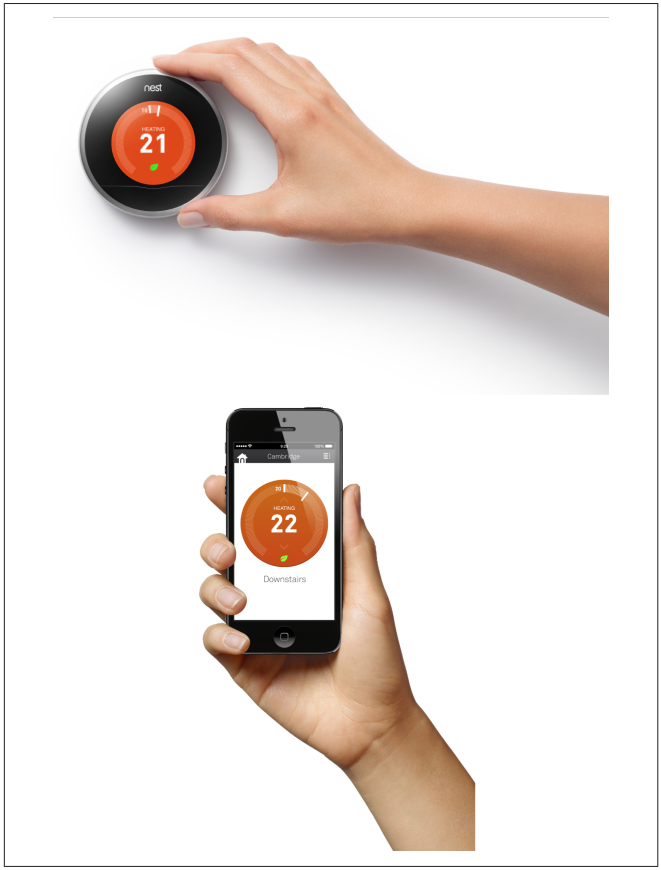

Figure 1-15. The Nest thermostat's primary user input method is a rotating bezel. But the smartphone app is optimized for tapping: it does not simulate dragging a fake bezel. The thermostat and app use complementary visual styling—tapping the smartphone interface produces the same subtle "click" as rotating the bezel on the physical device. This helps the interfaces feel like part of the same family (images: Nest).

Intermittent connectivity also affects IoT UX. In conventional UX, we assume devices are mostly connected, but many IoT devices may spend more time offline. As discussed before, some IoT devices have batteries and only connect intermittently to conserve power. They may take time to respond to instructions, leading different devices

in the system to appear briefly out of sync (see Figure 1-17). Or they may only report data at fixed intervals, so data is not always "live."

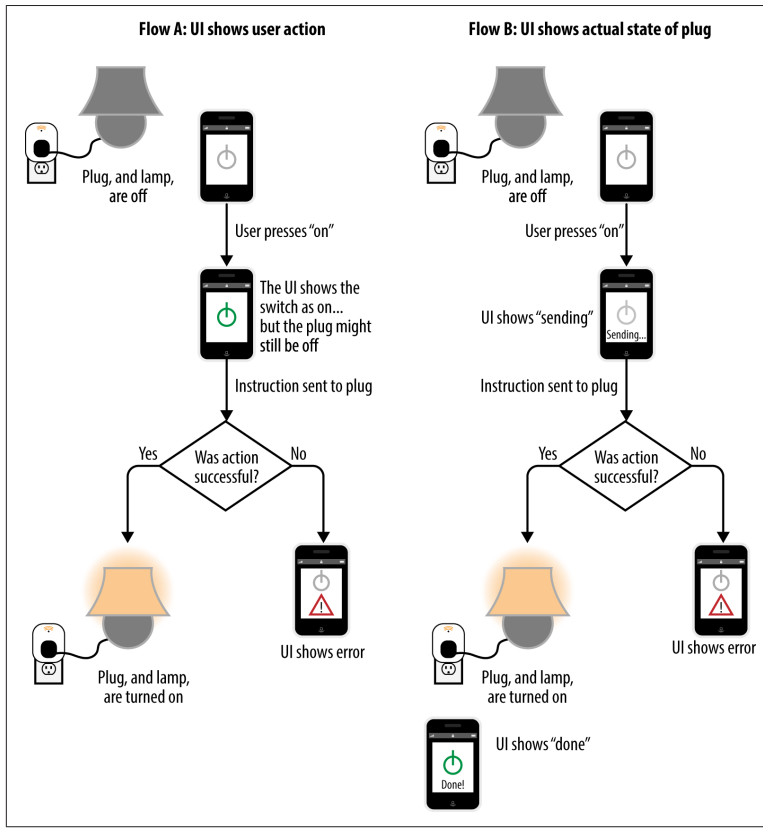

Figure 1-16. Two options for handling delays and potential failures for a smart plug connected to a lamp. In the "optimistic" example on the left, the UI confirms the user's action and backpedals if there is a problem. In the "pessimistic" example on the right, the UI confirms that the command is being sent to the plug but does not confirm that the plug has been turned on until it receives confirmation. The best approach for any given product will depend on the context of use.

Conceptual Model

The *conceptual model* is the understanding and expectations the user has of the system. What components does it have, how does it work, and how can they interact with it?

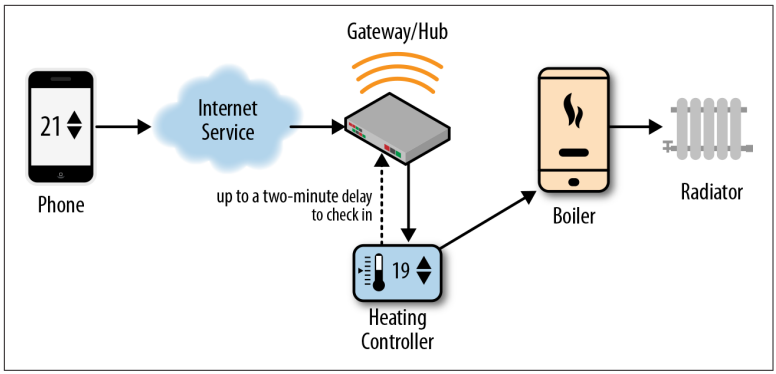

Figure 1-17. In this example, a battery-powered heating controller only checks into the network every two minutes for updated instructions. If the user turns up the heat from a smartphone app, there may be a delay of up to two minutes before the controller responds. During this time, the devices may report different status information about what the system is doing.

Non-connected products are often conceptually quite simple. A traditional lamp has a bulb, fitting, and switch. When the switch is flipped, electricity flows and the bulb lights up.

Connected products are more complex. There are often extra components, such as hubs or gateways. These infrastructure devices can seem obscure to mainstream consumers. There are more components to the system that require independent power and connectivity, and more places where code can run. Connected products are not objects, but systems.

When all the parts of the system are connected and working, users may not need to worry about which bit does what. But the system creates new ways for things to fail: if any part loses power or connectivity, the functioning of the system will be affected. Exactly what the impact is depends on the way the system is designed, and which component does what (see Figure 1-18).

In order to use a system effectively, users must have some understanding of what the different components do, and how they relate to each other.

For example, connected lighting systems often allow users to create automated rules that turn lights on and off at certain times or in response to triggers. Will those rules still run if the home Internet

connection goes down? This depends on where the code governing those rules runs. If it's on the user's phone or in the cloud, then it won't. If it's in a local hub, it will.

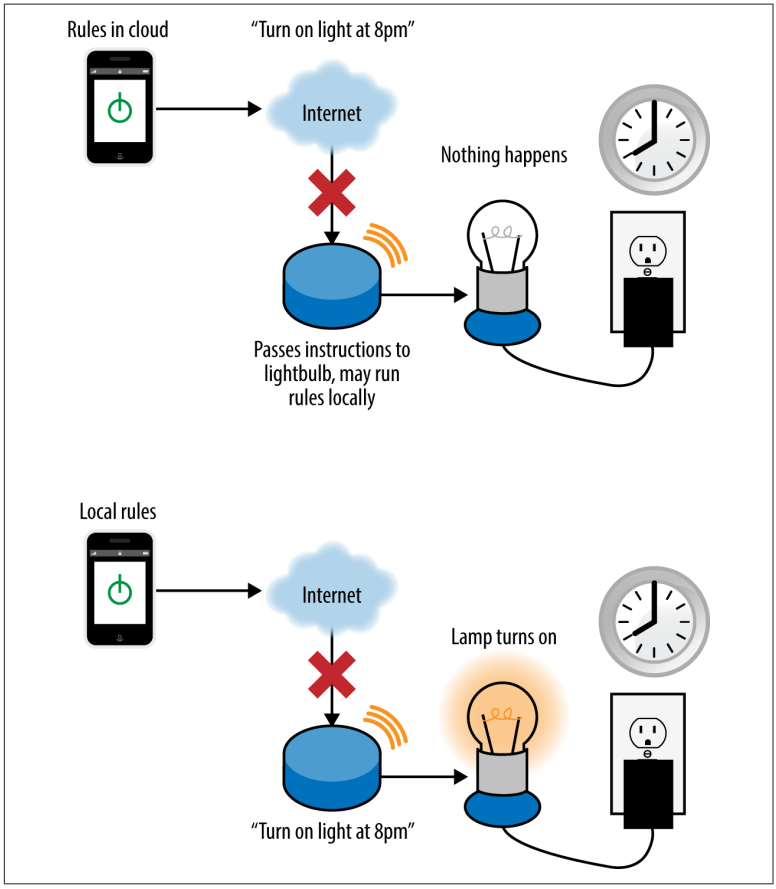

Figure 1-18. Connected home systems often offer automation rules, such as turning a light off at a specific time. If these are stored in the cloud, they will not run if the home Internet connection goes down. If they are stored locally, they will continue to run, but the user won't be able to see this or control devices remotely.

Helping users handle this extra complexity is not just a case of communicating how the system works. Designers must explicitly design clear conceptual models, taking into account users' existing knowledge, behaviors, and beliefs. The model is then communicated via system and interaction design and supporting documentation.

There are two key approaches. The first is to make the system's functioning very transparent (see Figure 1-19). The second is to simplify away the complexity (see Figure 1-20). As ever, the most appropriate choice depends on what the service does, and the context of use.

Figure 1-19. The BERG cloud bridge makes the status of network communications transparent: it is communicating the system model (image: BERG).

Service Design

Once purchased, a non-connected product may involve no further interaction with the vendor or manufacturer.

But a connected product is rarely a one-off purchase. It comes with the expectation of an ongoing service relationship with the vendor. At the very least, the user will expect the vendor to provide the Internet service that keeps it running, and to offer customer support. All of this forms part of the user's overall experience with the product. Products are no longer just fixed, physical things: they are combinations of hardware and software services. The devices themselves may be relatively dumb touchpoints for a dynamic and complex web service (the designer Mike Kuniavsky calls these "service avatars"). There may be offline service components, too.

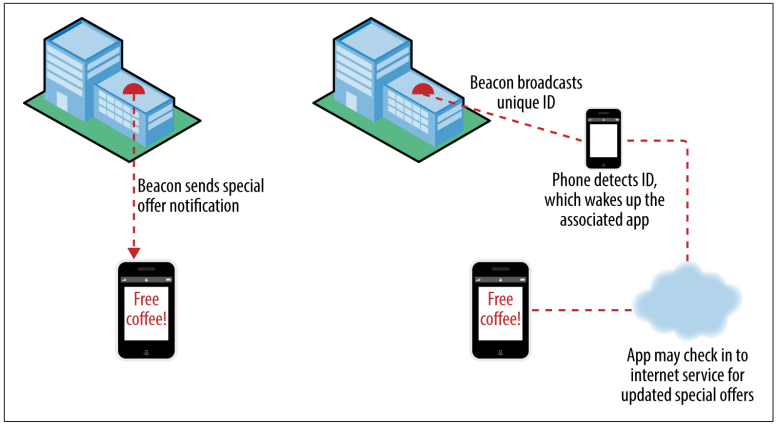

Figure 1-20. iBeacons operate with a simplified conceptual model. The user only needs to know that, for example, a store knows when they are nearby and can push special offers. They may think that the beacon itself is pushing out the offers. What actually happens is a little more complex, but the simplified conceptual model is good enough to use the system effectively.

Service design is an emerging field that takes a strategic, holistic view of UX. It considers how to deliver a coherent UX across all customer touchpoints (interfaces and offline components). It takes into account the user's changing needs over the whole lifespan of their relationship with the service. Its methods are not unique to IoT and can be applied to all kinds of online and offline services. But they are particularly well suited to thinking about ecosystems of devices, and systems that adapt and change over time.

The service around a connected product might include online and offline components such as:

- Continuity and reliability of the Internet service (what happens if the original supplier is acquired or goes out of business?)
- Marketing or sales materials
- Professional installation and servicing
- Instructional guides
- Customer support interactions
- In-store experiences
- Email communications and notifications

- Software updates and rollouts of new functionality
- Interoperating products or digital services from third-party suppliers
- Support for environmentally responsible disposal and recycling

Services are delivered through the interactions of networks of people, organizations, infrastructure, and physical components. Service design takes complete view of all the parts of your service, the stakeholders who are responsible for delivering them, and the relationships between them. The design specifies the interactions users should have with, and across, each touchpoint. It also designs the processes for ensuring that all the necessary components coordinate so the user is guaranteed to have the intended experience.

Creating a well-designed product is about getting the entire experience right, not just about the UI layer typically associated with design.

Productization

Productization is the activity of turning a concept or technology tool into a commercial product. In terms of design, the most significant part of this is defining a compelling product proposition. Who is the audience for the product? What value does the product provide for them? What core features and experiential requirements must be fulfilled to deliver that value? What is its business model?

Some of these things fall outside the traditional domain of UX designer, but they provide the essential underpinnings of good UX. 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.

A clear proposition doesn't just help users decide whether to buy the product in the first place. It also helps frame their mental model of the system and what it does. 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). See Figure 1-21.

The need for a clear value proposition is, of course, not unique to IoT, but has been an ongoing challenge. The consumer IoT product market is at a point of technological and market experimentation. This is good and healthy, but means that sometimes market fit, or

the clarity of propositions, has been overlooked in the gold rush of new technology. Some things are arguably being connected just because it's possible to do so, not because they need to be.

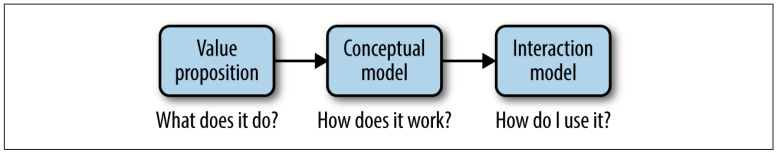

Figure 1-21. A good clear value proposition is fundamental to a great UX

This is less of a problem for products that target innovators and early adopters. These users are inherently interested in technology. They are often forgiving of imperfections and often don't mind spending time configuring and tinkering with the product—in fact, they usually enjoy doing so.

But mass-market consumers often have different needs. 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.

Many IoT systems are tools: they require the user to frame their own problem and configure the system to solve it. For example, connected electrical outlets/smart plugs are tools (see Figure 1-22). They can be used for a wide range of uses: in practice, many are put on lamps, but they can be used to control any device. Connected home monitoring kits comprised of a mix of sensors are also tools, as are smart rules systems like IFTTT.

Tools are powerful, because they can do many things. Early adopters may love them. But getting value from them requires an imaginative leap. Users must frame their own problem, identify the tool as a way to solve it, and configure it to meet their need.

Mainstream 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. The Nest Protect is a product. The advertising leads with all the ways in which the Protect is a great smoke/carbon monoxide alarm. It barely mentions connectiv-

ity, except to say that alerts will be sent to the user's phone. The Crock-Pot Slow Cooker with WeMo (see Figure 1-22) is also a product. Connectivity is a good fit for the context of use of a slow cooker. It alleviates the nervousness some people feel leaving a cooking appliance unattended at home all day. It allows users to adjust the cooking time remotely if they're going to be late (likely, as slow cookers are aimed at those with busy lifestyles).

Figure 1-22. Smart outlets, like the Belkin WeMo switch, are tools; the Crock-Pot Slow Cooker with WeMo is a product (image: Belkin).

Most people are busy, and have limited time and attention for configuring new products. Even if they are technically capable, they may have other, more pressing concerns that lead them to prefer a low-effort product to do the job instead.

Finally, business models shape the way users perceive the value of the service, fairness of pricing, and thus the UX. 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.

Putting connectivity and intelligence into devices may lead to digital business models appearing in the physical world. We may be offered devices cheaply in exchange for providing user data, or allowing our behavior to be monitored. Your supermarket might offer you a cheap fridge in exchange for monitoring the food you store in it (and where you buy it from). A coffee machine might notify the manufacturer when you are running low on pods, enabling a subscription model for consumables. Car insurers can set premiums

based on actual measured driver behavior rather than crude demographics.

Platform Design

Platform design refers to the technology enablers of the product, particularly those that enable multiple products to coordinate in sensible ways.

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.

Platforms may incorporate:

- Data (e.g., from sensors)
- Data models (e.g., for system coordination)
- APIs to build the UIs on top and to enable interconnections with other systems, devices, and services

Engineers will need to make many technical decisions during the creation of a product, many of which may not directly have an impact on the UX. But technology underpins and shapes the UX. It can enable amazing products and experiences, or drastically limit a products' ability to meet user needs. Engineers and designers need to collaborate to ensure that the system supports the right product features and the right design.

For example, the platform for a product like Philips 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 core point of collaboration between designers and engineers is ensuring a good fit between API and UX design. 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 there's no API for the feature or data point the designer wants, it can't be made to work in the UI. And if the API design is a poor fit for the requirements of the UX, it can result in a slow, unresponsive experience. This can happen when the API structure is not well aligned with the UX, or when too many API calls are needed for a specific screen or interaction.

APIs are also one of the enablers of interoperability: allowing systems to share data with other systems. Interoperability requires devices to support common network protocols (or to communicate via a gateway that bridges different network types). It also requires them to "speak a common language" in terms of data formats. Right now, there are major technical barriers to widespread interoperability in the IoT and many products do not interoperate at all, although the situation is improving. Designers will need to consider which, if any, third-party products can interoperate with the system and how they need to be accommodated in the UX.

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, she might reasonably expect the system to offer to add it to existing lighting controls and perhaps offer it as part of the security system. It's not important to make it talk to the toaster. This is common sense to a human. But the system won't know those things unless this kind of logic is encoded in the platform in the form of data and domain models.

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, and what APIs are needed. But once your system has multiple, interconnected devices, there will be design challenges that will require more complex platform logic to solve.

There is no commonly understood set of activities for this yet, but designers and engineers should collaborate to identify and shape the key platform issues that support good higher-level UX.

Summary

IoT design spans distributed systems of devices, situated in all the complexity and mess of the real world. This adds rich new layers of complex challenges to designing for connected products, as compared to most conventional UX.

UI design and industrial design are important, but only part of the overall experience. Users want products that offer clear solutions to problems, come with good service and support provision, and feel coherent to use across all touchpoints.

It's more important than ever to think of the user experience at a systems level, spanning user interactions with multiple devices, physical hardware, the properties of networks, wider business and service context, and the underlying technology enablers. The UX is not just the responsibility of the designers, but everyone involved, including product strategy and engineering.

About the Authors

Claire Rowland is an independent research, UX strategy, design, and product consultant focused on the consumer Internet of Things, and lead author of *Designing Connected Products: UX for the Consumer Internet of Things* (O'Reilly). She has particular expertise in connected home technology and energy management. She was previously the service design manager for AlertMe.com, a connected home platform (now owned by British Gas), and head of research for design agency Fjord.

Martin Charlier is an independent design consultant based in London and a coauthor of *Designing Connected Products: UX for the Consumer Internet of Things* (O'Reilly). He is also a cofounder of rain cloud, a research project exploring ambient devices. Martin's expertise goes across strategy, UX, and service and product design. He has previously worked at design agency Fjord, innovation firm Frog Design, and art collective Random International.