Chapter 12

DESIGN, PROTOTYPING, AND CONSTRUCTION

Objectives

The main goals of this chapter are to accomplish the following:

- Describe prototyping and the different types of prototyping activities.
- Enable you to produce simple prototypes from the models developed during the requirements activity.
- Enable you to produce a conceptual model for a product and justify your choices.
- Explain the use of scenarios and prototypes in design.
- Introduce both physical computing kits and software development kits and their role in construction.

12.1 Introduction

Design, prototyping, and construction fall within the Develop phase of the double diamond of design, introduced in Chapter 2, "The Process of Interaction Design," in which solutions or concepts are created, prototyped, tested, and iterated. The final product emerges iteratively through repeated design-evaluation-redesign cycles involving users, and prototypes facilitate this process. There are two aspects to design: the conceptual part, which focuses on the idea of a product, and the concrete aspect, which focuses on the details of the design. The former involves developing a conceptual model that captures what the product will do and how it will behave, while the latter is concerned with the details of the design, such as menu

types, haptic feedback, physical widgets, and graphics. The two are intertwined, and concrete design issues will require some consideration in order to prototype ideas, and prototyping ideas will lead to an evolution of the concept.

For users to evaluate the design of an interactive product effectively, designers prototype their ideas. In the early stages of development, these prototypes may be made of paper and cardboard, or ready-made components pulled together to allow evaluation, while as the design progresses, they become more polished, tailored, and robust so that they resemble the final product.

This chapter presents the activities involved in progressing a set of requirements through the cycles of prototyping and construction. The next section explains the role and techniques of prototyping and then explores how prototypes may be used in the design process. The chapter ends by discussing physical computing and software development kits (SDKs), which provide a basis for construction.

12.2 Prototyping

It is often said that users can't tell you what they want, but when they see something and get to use it, they soon know what they don't want. *Prototyping* provides a concrete manifestation of an idea—whether it is a new product or a modification of an existing one—which allows designers to communicate their ideas and users to try them out.

12.2.1 What Is a Prototype?

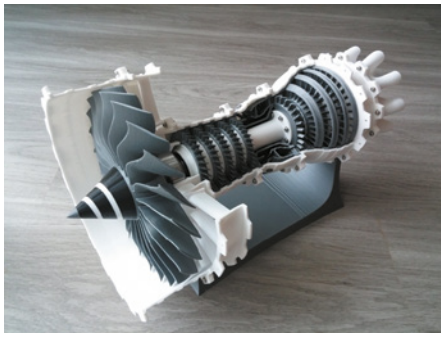
A *prototype* is one manifestation of a design that allows stakeholders to interact with it and to explore its suitability. It is limited in that a prototype will usually emphasize one set of product characteristics and de-emphasize others (see Box 12.1). Prototypes take many forms, for example, a scale model of a building or a bridge, or a piece of software that crashes every few minutes. A prototype can also be a paper-based outline of a display, a collection of wires and ready-made components, a digital picture, a video simulation, a complex piece of software and hardware, or a three-dimensional mockup of a workstation.

In fact, a prototype can be anything from a paper-based storyboard to a complex piece of software, and from a cardboard mockup to a molded or pressed piece of metal. For example, when the idea for the PalmPilot (a line of palmtop computers introduced in 1992) was being developed, Jeff Hawkin (founder of the company) carved up a piece of wood about the size and shape of the device he had imagined (see Figure 12.1).

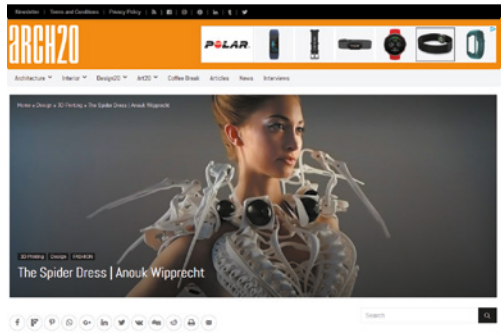
Jeff Hawkin used to carry this piece of wood around with him and pretend to enter information into it, just to see what it would be like to own such a device (Bergman and Haitani, 2000). This is an example of a simple (some might even say bizarre) prototype, but it served its purpose of simulating scenarios of use. Advances in 3D printer technologies, coupled with reduced prices, have increased their use in design. It is now common practice to take a 3D model from a software package and print a prototype. Soft toys, chocolate, dresses, and whole houses may be "printed" in this way (see Figure 12.2 and the following links).

Figure 12.1 The PalmPilot wooden prototype

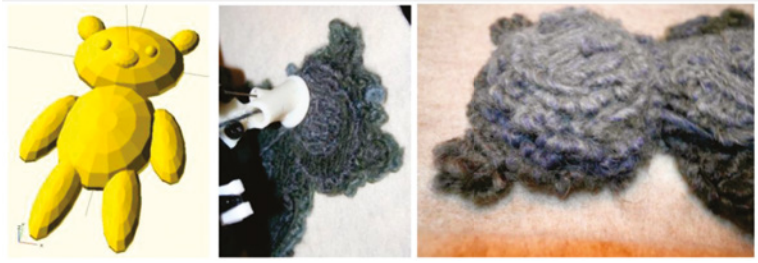
Source: https://www.computerhistory.org/revolution/mobile-computing/18/321/1648. © Mark Richards



(a)



(b)



(c)

Figure 12.2 Examples of 3D printing (a) model jet engine, (b) Spider Dress 2.0 by Anouk Wipprecht: embedded with sensors, the arms of the 'spider' will extend to defend the wearer if her breath becomes heavier, and (c) a teddy bear "printed" from a wireframe design

Source: (a) https://www.thingiverse.com/thing:392115. Licensed under CC-BY-3.0, (b) http://www.arch2o.com, and (c) Used courtesy of Scott Hudson

A video that shows how soft interactive objects can be printed is available at **https://www.youtube.com/watch?v=8jErWRddFYs**.

To see how 3D printing is facilitating fashion and interactive wearables, see the following articles:
https://interestingengineering.com/high-fashion-meets-3d-printing-9-3d-printed-dresses-for-the-future
https://medium.com/@scientiffic/designing-interactive-3d-printed-things-with-tinkercad-circuit-assemblies-518ee516adb6

12.2.2 Why Prototype?

Prototypes are useful when discussing or evaluating ideas with stakeholders; they are a communication device among team members and an effective way for designers to explore design ideas. The activity of building prototypes encourages reflection in design, as described by Donald Schön (1983), and it is recognized by designers from many disciplines as an important aspect of design.

Prototypes answer questions and support designers in choosing between alternatives. Hence, they serve a variety of purposes, for example, to test the technical feasibility of an idea, to clarify some vague requirements, to do some user testing and evaluation, or to check that a certain design direction is compatible with the rest of product development. The purpose of a prototype will influence the kind of prototype that is appropriate to build. So, for example, to clarify how users might perform a set of tasks and whether the proposed design would support them in doing this, a paper-based mockup might be produced. Figure 12.3 shows a paper-based prototype of a handheld device to help an autistic child communicate. This prototype shows the intended functions and buttons, their positioning and labeling, and the overall shape of the device, but none of the buttons actually works. This kind of prototype is sufficient to investigate scenarios of use and to decide, for example, whether the button images and labels are appropriate and the functions sufficient, but not to test whether the speech is loud enough or the response fast enough. Another example is the development of Halo, a new air quality monitor that can detect 10 different allergens and connects to an air purifier that will remove them (see the following reference). The design of Halo used a range of prototypes including many sketches (paper-based and electronic) and working prototypes.

Dan Saffer (2010) distinguishes between a product prototype and a service prototype, where the latter involves role-playing and people as an integral part of the prototype as well as the product itself. Service prototypes are sometimes captured as video scenarios and used in a similar way to the scenarios introduced in Chapter 11, "Discovering Requirements."

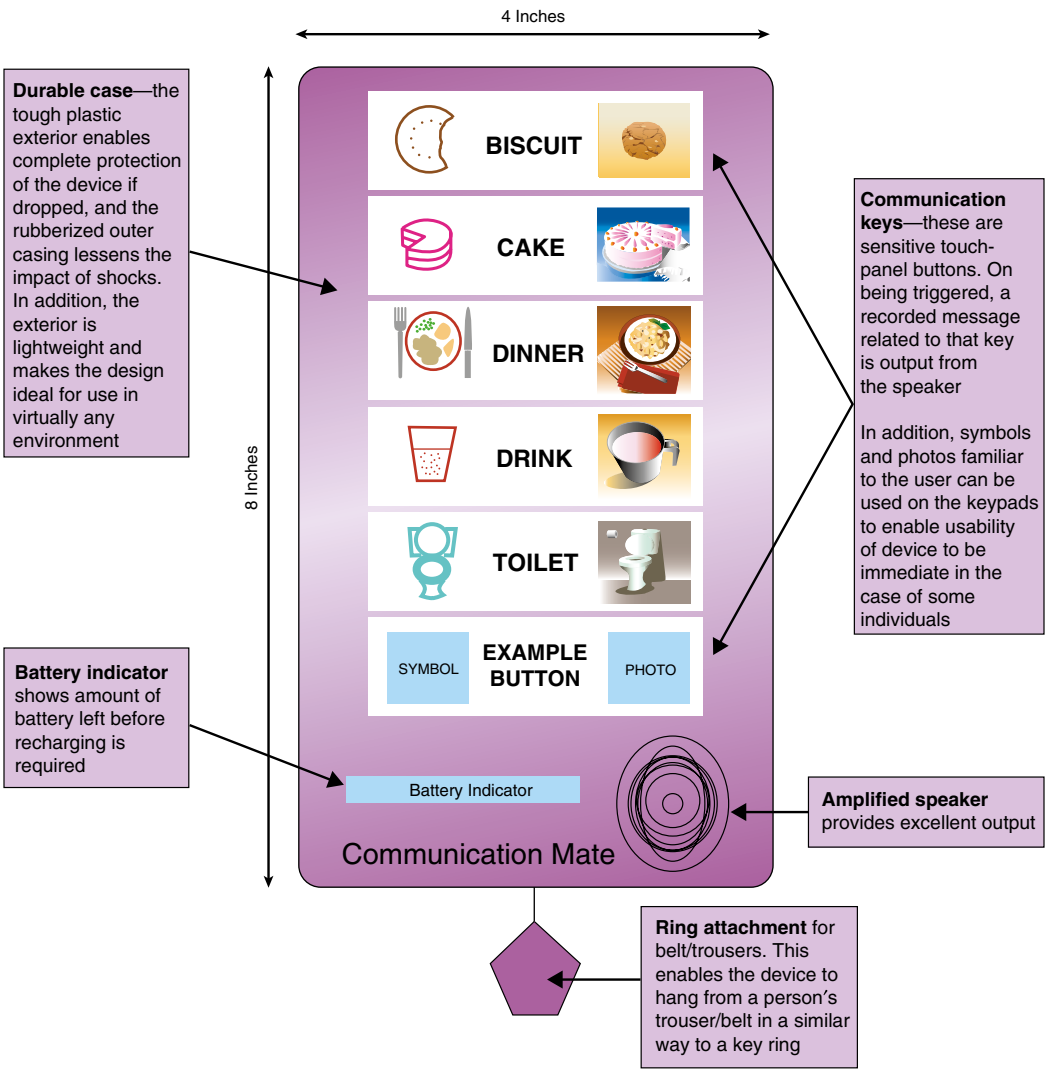


Figure 12.3 A paper-based prototype of a handheld device to support an autistic child
Source: Used courtesy of Sigil Khwaja

To see more about the development of the Wynd Halo and Wynd Home Purifier, see this website:
https://www.kickstarter.com/projects/882633450/wynd-halo-home-purifier-keep-your-homes-air-health?ref=discovery

12.2.3 Low-Fidelity Prototyping

A *low-fidelity prototype* does not look very much like the final product, nor does it provide the same functionality. For example, it may use very different materials, such as paper and cardboard rather than electronic screens and metal, it may perform only a limited set of functions, or it may only represent the functions and not perform any of them. The lump of wood used to prototype the PalmPilot described earlier is a low-fidelity prototype.

Low-fidelity prototypes are useful because they tend to be simple, cheap, and quick to produce. This also means that they are simple, cheap, and quick to modify so that they support the exploration of alternative designs and ideas. This is particularly important in the early stages of development, during conceptual design for example, because prototypes that are used for exploring ideas should be flexible and encourage exploration and modification. Low-fidelity prototypes are not meant to be kept and integrated into the final product.

Low-fidelity prototyping has other uses, for example in education. Seobkin Kang et al. (2018) use low-fidelity prototyping to help children represent creative ideas when designing and experimenting with complex systems. Their system, called *Rainbow*, is composed of a collection of low-fidelity materials such as paper, scissors, and markers that can be used to create prototypes, a top-down camera that can recognize the prototype, and a monitor to display augmented reality visualizations.

Storyboarding

Storyboarding is one example of low-fidelity prototyping that is often used in conjunction with scenarios, as described in Chapter 11, "Discovering Requirements." A *storyboard* consists of a series of sketches showing how a user might progress through a task using the product under development. It can be a series of screen sketches or a series of scenes showing how a user can perform a task using an interactive device. When used in conjunction with a scenario, the storyboard provides more detail and offers stakeholders a chance to role-play with a prototype, interacting with it by stepping through the scenario. The example storyboard shown in Figure 12.4 depicts a person (Christina) using a new mobile device for exploring historical

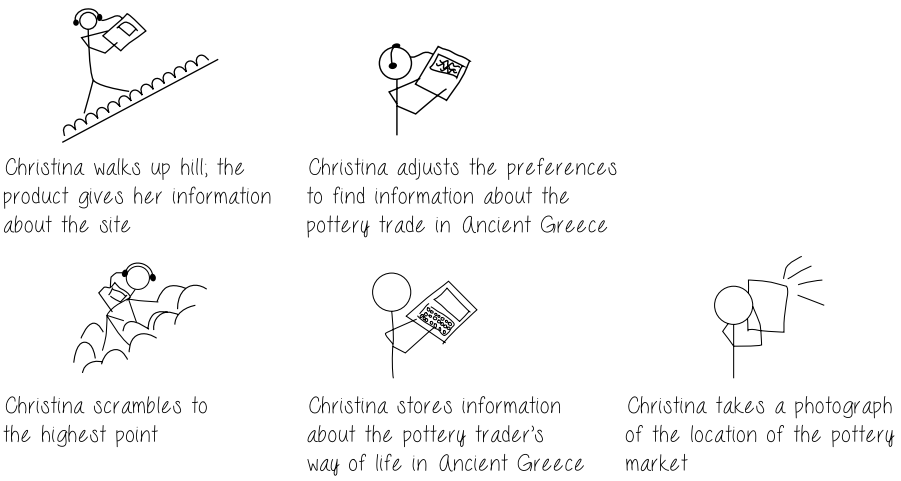


Figure 12.4 An example storyboard for a mobile device to explore ancient sites such as The Acropolis

sites. This example shows the context of use for this device and how it might support Christina in her quest for information about the pottery trade at The Acropolis in Ancient Greece.

Sketching

Low-fidelity prototyping often relies on hand-drawn sketches. Many people find it difficult to engage in *sketching* because they are inhibited by the quality of their drawing. As Saul Greenberg et al. (2012) put it, however, "Sketching is not about drawing. Rather, it is about design" (p. 7). You can get over any inhibition by devising your own symbols and icons and practicing them—referred to by Saul Greenberg et al. as a *sketching vocabulary* (p. 85). They don't have to be anything more than simple boxes, stick figures, and stars. Elements that might be required in a storyboard sketch, for example, include digital devices, people, emotions, tables, books, and so forth, and actions such as give, find, transfer, and write. If you are sketching an interface design, then you might need to draw various icons, dialog boxes, and so on. Some simple examples are shown in Figure 12.5. The next activity requires other sketching symbols, but they can still be drawn quite simply. Mark Baskinger (2008) provides further tips for those new to sketching.

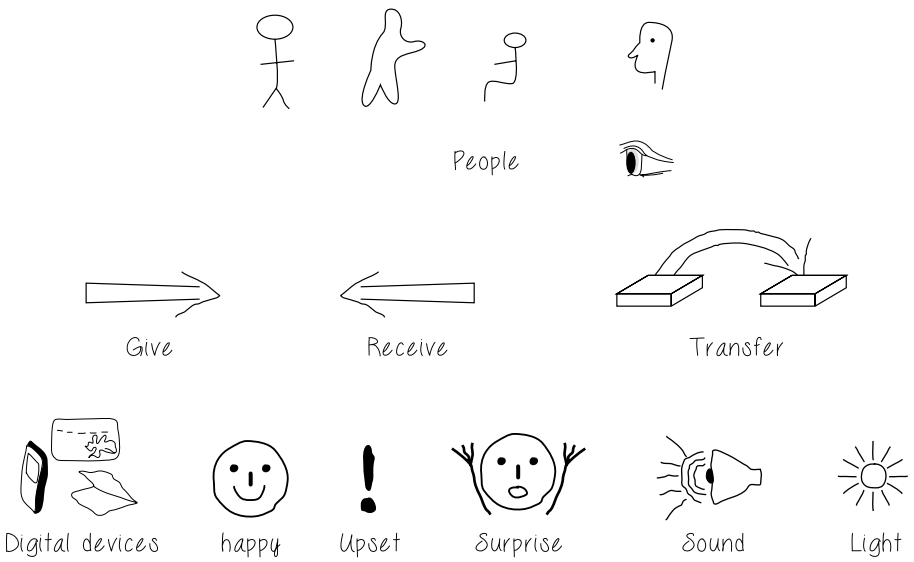


Figure 12.5 Some simple sketches for low-fidelity prototyping

Prototyping with Index Cards

Using *index cards* (small pieces of cardboard about 3 × 5 inches) is a successful and simple way to prototype an interaction, and it is used for developing a range of interactive products including websites and smartphone apps. Each card represents one element of the interaction, perhaps a screen or just an icon, menu, or dialog exchange. In user evaluations, the user can step through the cards, pretending to perform the task while interacting with the cards. A more detailed example of this kind of prototyping is provided in Section 12.5.2.

ACTIVITY 12.1

Produce a storyboard that depicts how to fill a car with fuel.

Comment

Our attempt is shown in Figure 12.6.

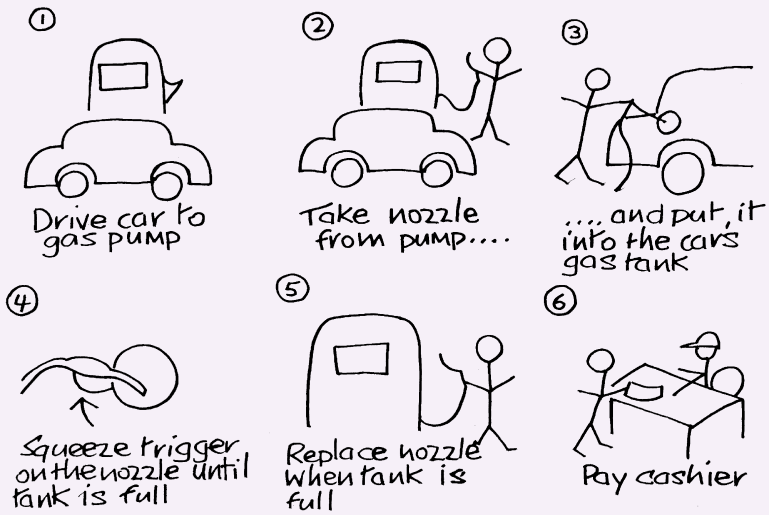


Figure 12.6 A storyboard depicting how to fill a car with fuel ■

Wizard of Oz

Another low-fidelity prototyping method called *Wizard of Oz* assumes that you have a software-based prototype. With this technique, the user interacts with the software as though interacting with the product. In fact, however, a human operator simulates the software's response to the user. The method takes its name from the classic story of the little girl who is swept away in a storm and finds herself in the Land of Oz (Baum and Denslow, 1900). The Wizard of Oz is a small shy man who operates a large artificial image of himself from behind a screen where no one can see him. The Wizard of Oz style of prototyping has been used successfully for various applications, including analyzing gestural behavior (Henschke et al., 2015) and when studying dialogues between children and virtual agents (Fialho and Coheur, 2015). The Wizard of Oz technique is often used in human-robot interaction studies. One such example is *Marionette*, a Wizard of Oz system for performing studies on the road with autonomous vehicles (Wang et al., 2017). Prototyping AI systems also draws on this style of prototyping, where the designer sketches the AI for themselves, and as the design matures, implementations of the AI can take its place (van Allen, 2018).

12.2.4 High-Fidelity Prototyping

A *high-fidelity prototype* looks more like the final product and usually provides more functionality than a low-fidelity prototype. For example, a prototype of a software system developed in Python or other executable language is higher fidelity than a paper-based mock-up; a

molded piece of plastic with a dummy keyboard would be a higher-fidelity prototype of the PalmPilot than the lump of wood. There is a continuum between low- and high-fidelity, and prototypes used "in the wild," for example, will have enough fidelity to be able to answer their design questions and to learn about interaction or technological constraints or contextual factors. It is common for prototypes to evolve through various stages of fidelity, within the design-evaluate-redesign cycles. Boban Blazevski and Jean Haslwanter (2017) describe their successful trials of two fully working prototypes for an assembly line mobile worker assistance system. They developed a smartphone app and a tablet-based app, both of which were integrated into the production system so that suitable instructions could be provided. Workers used the two versions for five days, and this allowed them to be evaluated *in situ*. The paper concludes that although producing a working prototype takes more effort, being able to try the prototype in real contexts provided valuable feedback for this kind of environment.

High-fidelity prototypes can be developed by modifying and integrating existing components—both hardware and software—which are widely available through various developer kits and open source software, for example. In robotics, this approach has been called *tinkering* (Hendriks-Jansen, 1996), while in software development it has been referred to as *Opportunistic System Development* (Ncube et al, 2008). For example, Ali Al-Humairi et al. (2018) used existing hardware (Arduino) and open source software to build a prototype to test their idea of robotically playing musical instruments automatically from a mobile phone.

12.2.5 Compromises in Prototyping

By their very nature, prototypes involve compromises: the intention is to produce something quickly to test an aspect of the product. Youn-Kiung Lim et al. (2008) suggest an anatomy of prototyping that structures the different aspects of a prototype and what it aims to achieve. Their ideas are expanded in Box 12.1. The kind of questions that any one prototype can answer is limited, and the prototype must be built with the key issues in mind. In low-fidelity prototyping, it is fairly clear that compromises have been made. For example, with a paper-based prototype, an obvious compromise is that the device doesn't actually work. For physical prototypes or software prototypes, some of the compromises will still be fairly clear. For example, the casing may not be very robust, the response speed may be slow, the look and feel may not be finalized, or only a limited amount of functionality may be available. Box 12.2 discusses when to use low- or high-fidelity prototyping.

BOX 12.1

The Anatomy of Prototyping: Filters and Manifestations

Youn-Kyung Lim et al. (2008) propose a view of prototypes that focuses on their role as filters, for example to emphasize specific aspects of a product being explored by the prototype, and as manifestations of designs, for instance, as tools to help designers develop their design ideas through external representations.

They suggest three key principles in their view of the anatomy of prototypes:

1. *Fundamental prototyping principle*: Prototyping is an activity with the purpose of creating a manifestation that, in its simplest form, filters the qualities in which designers are interested without distorting the understanding of the whole.

(*Continued*)

2. *Economic principle of prototyping*: 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.
3. *Anatomy of prototypes*: Prototypes are filters that traverse a design space and are manifestations of design ideas that concretize and externalize conceptual ideas.

Youn-Kyung Lim et al. identify several dimensions of filtering and of manifestation that may be considered when developing a prototype, although they point out that these dimensions are not complete but provide a useful starting point for consideration of prototype development. These are shown in Table 12.1 and Table 12.2.

Filtering dimension	Example variables
Appearance	size; color; shape; margin; form; weight; texture; proportion; hardness; transparency; gradation; haptic; sound
Data	data size; data type (for example, number; string; media); data use; privacy type; hierarchy; organization
Functionality	system function; users' functionality needs
Interactivity	input behavior; output behavior; feedback behavior; information behavior
Spatial structure	arrangement of interface or information elements; relationship among interface or information elements, which can be either two- or three-dimensional, intangible or tangible, or mixed

Table 12.1 Example variables of each filtering dimension

Manifestation dimension	Definition	Example variables
Material	Medium (either visible or invisible) used to form a prototype	Physical media, for example, paper, wood, and plastic; tools for manipulating physical matters, such as a knife, scissors, pen, and sandpaper; computational prototyping tools, for instance, Python; physical computing tools, such as, Phidgets and Basic Stamps; available existing artifacts, such as a beeper to simulate a heart attack
Resolution	Level of detail or sophistication of what is manifested (corresponding to fidelity)	Accuracy of performance, for instance, feedback time responding to an input by a user (giving user feedback in a paper prototype is slower than in a computer-based one); appearance details; interactivity details; realistic versus faked data
Scope	Range of what is covered to be manifested	Level of contextualization, for example, website color scheme testing with only color scheme charts or color schemes placed in a website layout structure; book search navigation usability testing with only the book search related interface or the whole navigation interface

Table 12.2 The definition and variables of each manifestation dimension ■

BOX 12.2

High-Fidelity and Low-Fidelity Prototypes

Table 12.3 summarizes proclaimed advantages and disadvantages of high- and low-fidelity prototyping.

Type	Advantages	Disadvantages
Low-fidelity prototype	• Quick revision possible • More time can be spent on improving the design before starting development • Evaluates multiple design concepts • Useful communication device • Proof of concept	• Limited error checking • Poor detailed specification for development • Facilitator-driven • Limited usefulness for usability tests • Navigational and flow limitations
High-fidelity prototype	• (Almost) complete functionality • Fully interactive • User-driven • Clearly defines navigational scheme • Use for exploration and test • Look and feel of intended product • Serves as a "living" or evolving specification • Marketing and sales tool	• More resource-intensive to develop • Time-consuming to modify • Inefficient for proof-of-concept designs • Potential of being mistaken for the final product • Potential of setting inappropriate expectations

Table 12.3 Advantages and disadvantages of low- and high-fidelity prototypes

Component kits and pattern libraries for interface components (see section 12.7 and Chapter 13, "Interaction Design in Practice") make it quite easy to develop polished functional prototypes quickly, but there is a strong case for the value of low-fidelity prototypes, such as paper-based sketches, sticky note designs, and storyboarding to explore initial ideas. Paper prototyping, for example, is used in game design (Gibson, 2014), website development, and product design (Case Study 12.1). Both high- and low-fidelity prototypes can provide useful feedback during evaluation and design iterations, but how do you know which to choose? The advantages and disadvantages listed earlier will help, but there are other factors too. Beant Dhillon et al. (2011) found that a low-fidelity video prototype elicited comparable user feedback as a high-fidelity one, but it was quicker and cheaper to produce. Gavin Sim et al. (2016) investigated the effect of using a paper booklet versus iPads with children who were rating a game concept. They found that the children's rating of the game was unaffected by the form factor but that they rated the paper version significantly higher for aesthetics. ■

This article covers the benefits of high- and low-fidelity prototyping and how to produce them:
https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/?lm=aesthetic-usability-effect&pt=article

CASE STUDY 12.1

Paper Prototyping as a Core Tool in the Design of Telephone User Interfaces

Paper prototyping is being used by telephone and tablet companies as a core part of their design process (see Figure 12.7). Mobile devices are feature-rich. They include megapixel cameras, music players, media galleries, downloaded applications, and more. This requires designing interactions that are complex but that are clear to learn and use. Paper prototyping offers a rapid way to work through every detail of the interaction design across multiple applications.

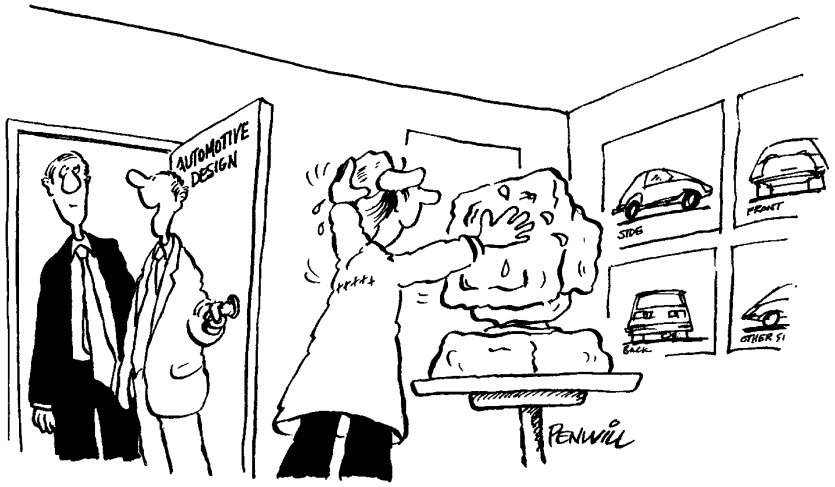
Figure 12.7 Prototype developed for cell phone user interface

Mobile device projects involve a range of disciplines—all with their own perspective on what the product should be. A typical project may include programmers, project managers, marketing experts, commercial managers, handset manufacturers, user experience specialists, visual designers, content managers, and network specialists. Paper prototyping provides a vehicle for everyone involved to be part of the design process—considering the design from multiple angles in a collaborative way.

The case study on the id-book.com website describes the benefits of using paper prototyping from the designer's viewpoint, while considering the bigger picture of its impact across the entire project lifecycle. It starts by explaining the problem space and how paper prototyping is used as an integrated part of user interface design projects for European and U.S.-based mobile operator companies. The case study uses project examples to illustrate the approach and explains step by step how the method can be used to include a range of stakeholders in the design process—regardless of their skill set or background. The case study offers exercises so that you can experiment with the approach yourself. ■

Two common properties that are often traded off against each other are breadth of functionality versus depth. These two kinds of prototyping are called *horizontal prototyping* (providing a wide range of functions but with little detail) and *vertical prototyping* (providing a lot of detail for only a few functions). Another common compromise is level of robustness versus degree of changeability. Making a prototype robust may lead to it being harder to change. This compromise may not be visible to a user of the product until something goes wrong. For example, the internal structure of a piece of code may not have been carefully designed, or the connections between electronic components may be delicate.

One of the consequences of high-fidelity prototypes is that the prototype can appear to be good enough to be the final product, and users may be less prepared to critique something if they perceive it as a finished product. Another consequence is that fewer alternatives are considered because the prototype works and users like it.



"THEN IN HERE WE DO A CLAY MOCK-UP OF THE COMPUTER MODEL"

Source: Reproduced with permission of Penwil Cartoons

Although prototypes will have undergone extensive user evaluation, they will not necessarily have been built with good engineering principles or subjected to rigorous quality testing for other characteristics such as security and error-free operation. Building a product to be used by thousands or millions of people running on various platforms and under a wide range of circumstances requires a different construction and testing regime than producing a quick prototype to answer specific questions.

The next "Dilemma" box discusses two different development philosophies. In *evolutionary prototyping*, a prototype evolves into the final product and is built with these engineering principles in mind. *Throwaway prototyping* uses the prototypes as stepping stones toward the final design. In this case, the prototypes are thrown away, and the final product is built from scratch. In an evolutionary prototyping approach, each stage will be subjected to rigorous testing; for throwaway prototyping, such testing is not necessary.

DILEMMA

Prototyping vs. Engineering

The compromises made when developing low-fidelity prototypes are evident, but compromises in high-fidelity prototypes are not so obvious. When a project is under pressure, it can become tempting to integrate a set of existing high-fidelity prototypes together to form the final product. Many hours will have been spent developing them, and evaluation with users has gone well. So, why throw it all away? Generating the final product this way will simply store up testing and maintenance problems for later (see Box 13.1 on technical debt). In short, this is likely to compromise the quality of the product, unless the prototypes have been built with sound engineering principles from the start.

On the other hand, if the device is an innovation, then being first to market with a "good enough" product may be more important for securing market position than having a very high-quality product that reaches the market two months after a competitor's product.

The dilemma arises in deciding how to treat high-fidelity prototypes—engineer them from the start or accept that they will be thrown away. ■

12.3 Conceptual Design

Conceptual design is concerned with developing a conceptual model; conceptual models were introduced in Chapter 3, "Conceptualizing Interaction." The idea of a conceptual model can be difficult to grasp because these models take many different forms and there isn't a definitive detailed characterization of one. Instead, conceptual design is best understood by exploring and experiencing different approaches to it, and the purpose of this section is to provide some concrete suggestions about how to go about doing this.

A *conceptual model* is an outline of what people can do with a product and which concepts are needed for the user to understand how to interact with it. The former will emerge from an understanding of the problem space and the current functional requirements. Which concepts are needed to understand how to interact with the product depends on a variety of issues such as who the user will be, what kind of interaction will be used, what kind of interface will be used, terminology, metaphors, application domain, and so on? The first step in developing a conceptual model is to steep yourself in the data about the users and their goals and try to empathize with them.

There are different ways to achieve empathy with users. For example, Karen Holtzblatt and Hugh Beyer's (2017) contextual interviews, interpretation sessions and the Wall Walk, introduced in Chapter 11, supports this. These three activities together ensure that different people's perspectives on the data and what they observed are captured, help to deepen understanding and to expose the whole team to different aspects of the problem space, and immerse the team in the users' world. This stimulates ideas based on an extended understanding of the user and their context. Once captured, ideas are tested against other data and scenarios, discussed with

other design team members, and prototyped for testing with users. *Contextual Design* describes further activities that cover the whole design process. Trying to empathize with users may not be the right approach, however, as discussed in the following "Dilemma" box.

Using different creativity and brainstorming techniques to explore ideas with other members of the team can help build a picture of the users and their goals. Gradually, a picture of the desired users' experience will emerge and become more concrete. This process is helped by considering the issues in this section and by using scenarios and prototypes to capture and experiment with ideas. The availability of ready-made components increases the ease with which ideas can be prototyped, which also helps to explore different conceptual models and design ideas. Mood boards (traditionally used in fashion and interior design) may be used to capture the desired feel of a new product (see Figure 12.8). This is informed by any data gathering or evaluation activities and considered in the context of technological feasibility.

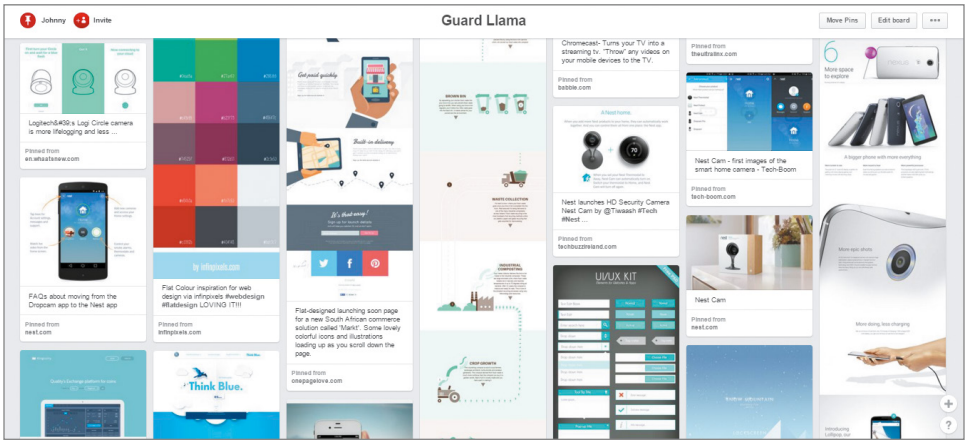


Figure 12.8 An example mood board developed for a personal safety product called Guard Llama
Source: http://johnnyhuang.design/guardllama.html

Read about how to create mood boards for UX projects here:
https://uxplanet.org/creating-better-moodboards-for-ux-projects-381d4d6daf70
Invision offers a tool to help with this. See the following web page:
https://www.invisionapp.com/inside-design/boards-share-design-inspiration-assets/

Developing a range of scenarios, as described in Chapter 11, can also help with conceptual design (Bødker, 2000) and to think through the consequences of different ideas. Suzanne Bødker (2000) also proposes the notion of plus and minus scenarios. These attempt to capture the most positive and the most negative consequences of a particular proposed design solution, thereby helping designers to gain a more comprehensive view of the proposal.

This idea has been extended by Mancini et al. (2010) who used positive and negative video scenarios to explore futuristic technology. Their approach used video to represent positive and negative consequences of a new product to help with diet and well-being, which was designed to explore privacy concerns and attitudes. The two videos (each with six scenes) focus on Peter, an overweight businessman who has been advised by his doctor to use a new product DietMon to help him lose weight. The product consists of glasses with a hidden camera, a microchip in the wrist, a central data store, and a text messaging system to send messages to Peter's mobile phone telling him the calorific value of the food he is looking at and warning him when he is close to his daily limit (Price et al, 2010). Figure 12.9 shows the content of two scenes from the videos and the positive and negative reactions; Figure 12.10 is a snapshot from the negative video.

Scene 2: breakfast at home	
Peter starts preparing his breakfast with his new glasses on. His wife notices them and he *keenly* gives her a demonstration of what they are and how they work, and tells her about the microchip. She seems *impressed* and leaves the room to get ready for work. Peter opens the fridge to put away the butter and sees a pastry. He looks at it and gets a DietMon message telling him the calorie content of the pastry. He shows that to his wife, who is entering the kitchen and looks at him with a *smile*.	Peter prepares breakfast with his new glasses on. His wife notices them. While looking at his toast, he gets a text. His wife enquires what that is. He says it's nothing and he does not feel like having toast after all. When she questions why he becomes *tense* and *reluctantly* tells her about DietMon. *Skeptical*, she leaves the room with a sarcastic comment. Peter opens the fridge and sees a pastry. As he gives in and takes a bite, he is caught by his wife, who is entering the kitchen and looks at him with a *grin*.
Scene 3: birthday party at the office	
Peter is working away at his desk when some colleagues invite him to a small birthday celebration. He tries to refuse but they insist. As he joins them, wearing his glasses, he greets the birthday-lady. His colleague Chris serves him a slice of cake. Peter looks at it and takes out his mobile. He gets a text, checks it and says the slice is too big, and asks Chris to cut it in a half. Chris is intrigued and asks for an explanation, so Peter gives his colleagues a *keen* demonstration of how the technology works. His audience is *impressed*, gathered around him.	Peter is working away at his desk when some colleagues invite him to a small birthday celebration. He tries to refuse but they insist. As he joins them, wearing his glasses, his colleague Chris gives him a slice of cake. He takes the plate and greets the birthday-lady. He gets a text and, *pretending* it's an important phone call, moves *away* from the others with the cake. Turned away from them, he *throws* the cake in a bin and goes back pretending to have already finished it. Chris comments on how fast he ate. Peter excuses himself, saying he has a deadline to meet, and leaves.

Figure 12.9 How two scenes from the videos differ in terms of positive and negative reactions to the system. The positive version is on the left and the negative on the right

Source: Mancini et al. (2010)

Figure 12.10 Peter being caught eating the pastry out of the fridge at breakfast (scene 2, negative reaction)

Source: Price et al. (2010)

DILEMMA

Is Trying to Empathize with Users the Right Approach?

Empathizing with users who live in a very different context than the designers is not easy, no matter how much data is collected. Interaction designers have tried several ways to understand situations that are outside their experience, two of which are experience prototyping and the Third Age suit.

(*Continued*)

Experience prototyping was introduced by Marion Buchenau and Jane Fulton Suri (2000) who describe a team designing a chest-implanted automatic defibrillator. A defibrillator is used with victims of cardiac arrest to deliver a strong electric shock to the heart that is intended to restore the heart muscle to its regular rhythm. This kind of event is completely outside most people's experience. To simulate some critical aspects of the experience, one of which is the random occurrence of a defibrillating shock, each team member was sent text messages at random times over one weekend. Each message simulated the occurrence of a defibrillating shock, and team members were asked to record where they were, who they were with, what they were doing, and what they thought and felt knowing that this represented a shock. Example insights ranged from anxiety around everyday happenings, such as holding a child and operating power tools, to being in social situations and at a loss how to communicate to onlookers what was happening. This first-hand experience brought new insights to the design effort.

The second example is the Third Age suit, designed so that car designers can experience what it is like for people with some loss of mobility or declining sensory perception to drive their cars. The suit restricts movement in the neck, arms, legs, and ankles. Originally developed by Ford Motor Company and Loughborough University (see Figure 12.11), it has been used to raise awareness within groups of car designers, architects, and other product designers.

Using these techniques appears to have brought new insights to the design process, but how deep are those insights and how accurate are they? According to Michelle Nario-Redmond

Figure 12.11 The Third Age suit helps designers experience the loss of mobility and sensory perception.

Source: Ford Motor Co.

et al. (2017), who conducted experiments to investigate the impact of disability simulations, they have unexpected negative consequences. They found that these simulations can result in feelings of fear, apprehension, and pity toward those with disabilities, rather than any sense of empathy. In addition, experiencing the disability for only a short time does not take into account the various coping strategies and innovative techniques that individuals develop. They suggest that a better way to design for these circumstances is to involve people with those disabilities and to understand their experiences more holistically.

To see the Third Age suit in action, watch this video:
https://www.youtube.com/watch?v=Yb0aqr0rzrs

To read an overview of the disability simulation experiment results, see this article:
https://blog.prototypr.io/why-i-wont-try-on-disability-to-build-empathy-in-the-design-process-and-you-should-think-twice-7086ed6202aa

12.3.1 Developing an Initial Conceptual Model

The core components of the conceptual model are metaphor and analogies, the concepts to which users are exposed, the relationship between those concepts, and the mappings between the concepts and user experience being supported (Chapter 3). Some of these will derive from the product's requirements, such as the concepts involved in a task and their relationships, such as through scenarios and use cases. Others such as suitable metaphors and analogies will be informed by immersion in the data and attempting to understand the users' perspectives.

This section introduces approaches that help to produce an initial conceptual model. In particular, it considers the following:

- How to choose interface metaphors that will help users understand the product?
- Which interaction type(s) would best support the users' activities?
- Do different interface types suggest alternative design insights or options?

All of these approaches provide different ways of thinking about the product and help generate potential conceptual models.

Interface Metaphors

Interface metaphors combine familiar knowledge with new knowledge in a way that will help users understand the product. Choosing suitable metaphors and combining new and familiar concepts requires a balance between utility and relevance, and it is based on an understanding of the users and their context. For example, consider an educational system to

teach 6-year-olds mathematics. One possible metaphor is a classroom with a teacher standing at the front. But considering the users of the product and what is likely to engage them, a metaphor that reminds them of something enjoyable is more likely to keep them engaged, such as a ball game, the circus, a playroom, and so on.

Different approaches to identifying and choosing an interface metaphor have been tried. For example, Dietrich Kammer et al. (2013) combined creativity methods to explore everyday objects, paper prototypes, and toolkits to support groups of students designing novel interface metaphors and gestures for mobile devices. They found that developing metaphors for both tablets and smartphones resulted in flexible metaphors. On the other hand, Marco Speicher et al. (2018) decided on an apartment metaphor for a VR online shopping experience by considering the limitations of systems that try to mimic physical stores.

Tom Erickson (1990) suggests a three-step process for choosing a good interface metaphor. Although this work is quite old, the approach is remarkably useful with current technologies. The first step is to understand what the system will do, that is, to identify functional requirements. Developing partial conceptual models and trying them may be part of the process. The second step is to understand which bits of the product are likely to cause users problems, that is, which tasks or subtasks cause problems, are complicated, or are critical. A metaphor is only a partial mapping between the product and the real thing upon which the metaphor is based. Understanding areas in which users are likely to have difficulties means that the metaphor can be chosen to support those aspects. The third step is to generate metaphors. Looking for metaphors in the users' description of relevant activities, or identifying metaphors used in the application domain, is a good starting point.

When suitable metaphors have been generated, they need to be evaluated. Erickson (1990) suggests five questions to ask:

- How much structure does the metaphor provide? A good metaphor will provide structure—preferably familiar structure.
- How much of the metaphor is relevant to the problem? One of the difficulties of using metaphors is that users may think they understand more than they do and start applying inappropriate elements of the metaphor to the product, leading to confusion or false expectations.
- Is the interface metaphor easy to represent? A good metaphor will be associated with particular physical, visual, and audio elements, as well as words.
- Will your audience understand the metaphor?
- How extensible is the metaphor? Does it have extra aspects that may be useful later?

For the group travel organizer introduced in Chapter 11, one potential metaphor that was prompted by the quote from Sky in her persona is a family restaurant. This seems appropriate because the family is all together, and each can choose what they want. Evaluating this metaphor using the previous five questions listed prompted the following thoughts:

- Does it supply structure? Yes, it supplies structure from the users' point of view, based on the familiar restaurant environment. Restaurants can be very different in their interior and the food they offer, but the structure includes having tables and a menu and people to serve the food. The experience of going to a restaurant involves arriving, sitting at a table, ordering food, being served the food, eating it, and then paying before leaving. From a different point of view, there is also structure around food preparation and how the kitchens are run.

- How much of the metaphor is relevant? Choosing a vacation involves seeing what is being offered and deciding what is most attractive, based on the preferences of everyone in the group. This is similar to choosing a meal in a restaurant. For example, a restaurant will have a menu, and visitors to the restaurant will sit together and choose individual meals, but they all sit in the same restaurant and enjoy the environment. For a group vacation, it may be that some members of the group want to do different activities and come together for some of the time, so this is similar. Information about the food such as allergens is available from the server or in the menu, reviews of restaurants are available, and photos or models of the food available are common. All of these characteristics are relevant to the group travel organizer. One of the characteristics of a restaurant that is not so applicable to a vacation is the need to pay at the end of the meal rather than before you get there.
- Is the metaphor easy to represent? There are several options in this regard, but the basic structure of a restaurant can be represented. The key aspect of this conceptual model will be to identify potential vacations that suit everyone and choose one. In a restaurant this process involves looking at menus, talking to the server, and ordering the food. Vacation information including photos and videos could be presented in a menu—maybe as one menu for adults and one for children. So, the main elements of the metaphor seem straightforward to represent.
- Will your audience understand the metaphor? For this example, the user group has not yet been investigated in detail, but eating in a restaurant is common.
- How extensible is the metaphor? There are several different types of restaurant experiences—à la carte, fixed menu, serve yourself, all you can eat, and food courts, for example. Elements from these different types of restaurants may be used to expand initial ideas.

ACTIVITY 12.2

One of the disadvantages of the restaurant metaphor is the need to have a shared experience when members of the group are in different locations. Another possible interface metaphor for the group travel organizer is the travel consultant. A travel consultant discusses the requirements with the traveler(s) and tailors the vacation accordingly, offering maybe two or three alternatives, but making most of the decisions on the travelers' behalf. Ask the earlier five questions about this metaphor.

Comment

1. Does the travel consultant metaphor supply structure?
   Yes.The key characteristic of this metaphor is that the travelers specify what they want, and the consultant researches the options. It relies on the travelers giving the consultant sufficient information to search within a suitable range rather than leaving them to make key decisions.
2. How much of the metaphor is relevant?
   The idea of handing over responsibility to someone else to search for suitable vacations may be appealing to some users, but it might feel uncomfortable to others. The level of

(*Continued*)

responsibility given to the consultant can be adjusted, though, depending on user preferences. It is common for individuals to put together vacations themselves based on web searches, but this can be time-consuming and diminish the excitement of planning a vacation. It would be attractive to some users if the initial searching and sifting is done for them.

3. Is the metaphor easy to represent?
   Yes, it could be represented by a software agent or by having a sophisticated database entry and search facility. But the question is, would users like this approach?
4. Will your audience understand the metaphor?
   Yes.
5. How extensible is the metaphor?
   The wonderful thing about people is that they are flexible; hence, the metaphor of the travel consultant is also pretty flexible. For example, the consultant could be asked to refine their vacation recommendations according to as many different criteria as the travelers require. ■

Interaction Types

Chapter 3 introduced five different types of interaction: instructing, conversing, manipulating, exploring, and responding. Which type of interaction is best suited to the current design depends on the application domain and the kind of product being developed. For example, a computer game is most likely to suit a manipulating style, while a software application for drawing or drafting has aspects of instructing and conversing.

Most conceptual models will include a combination of interaction types, and different parts of the interaction will be associated with different types. For example, in the group travel organizer, one of the user tasks is to find out the visa regulations for a particular destination. This will require an instructing approach to interaction as no dialog is necessary for the system to show the regulations. The user simply has to enter a predefined set of information, for instance, the country issuing the passport and destination. On the other hand, trying to identify a vacation for a group of people may be conducted more like a conversation. For example, the user may begin by selecting some characteristics of the destination and some time constraints and preferences. Then the organizer will respond with several options, and the user will provide more information or preferences and so on. Alternatively, for users who don't have any clear requirements yet, they might prefer to explore availability before asking for specific options. Responding could be used when the user chooses an option that has additional restrictions and the system asks if the user meets them.

Interface Types

Considering different interfaces at this stage may seem premature, but it has both a design and a practical purpose. When thinking about the conceptual model for a product, it is important not to be unduly influenced by a predetermined interface type. Different interface types prompt and support different perspectives on potential user experiences and possible behaviors, hence prompting alternative design ideas.

In practical terms, prototyping the product will require an interface type, or at least candidate alternative interface types. Which ones to choose depends on the product constraints

that arise from the requirements. For example, input and output modes will be influenced by user and environmental requirements. Therefore, considering interfaces at this point also takes one step toward producing practical prototypes.

To illustrate this, we consider a subset of the interfaces introduced in Chapter 7, "Interfaces," and the different perspectives they bring to the group travel organizer.

- **Shareable Interface** The travel organizer has to be shareable, as it is intended to be used by a group of people and it should be exciting and fun. The design issues for *shareable interfaces*, which were introduced in Chapter 7, will need to be considered for this system. For example, how best (whether) to use the individuals' own devices such as smartphones in conjunction with a shared interface. Allowing group members to interact at a distance suggests the need for multiple devices, so a combination of form factors is required.
- **Tangible Interface** *Tangible interfaces* are a kind of sensor-based interaction, where blocks or other physical objects are moved around. Thinking about a travel organizer in this way conjures up an interesting image of people collaborating, maybe with the physical objects representing themselves while traveling, but there are practical problems of having this kind of interface, as the objects may be lost or damaged.
- **Virtual Reality** The travel organizer seems to be an ideal product for making use of a *virtual reality* interface, as it would allow individuals to experience the destination and maybe some of the activities available. Virtual reality would not be needed for the whole product, just for the elements where users want to experience the destination.

ACTIVITY 12.3

Consider the new navigation app for a large shopping center introduced in Chapter 11.

1. Identify tasks associated with this product that would best be supported by each of the interaction types instructing, conversing, manipulating, exploring, and responding.
2. Pick out two interface types from Chapter 7 that might provide different perspectives on the design.

Comment

1. Here are some suggestions. You may have identified others.
   - **Instructing** The user wants to see the location of a specific shop.
   - **Conversing** The user wants to find one particular branch out of several; the app might ask them to pick one from a list. Or, the user might want to find a particular kind of shop, and the app will display a list from which to choose.
   - **Manipulating** The chosen route could be modified by dragging the path to encompass other shops or specific walkways.
   - **Exploring** The user might be able to walk around the shopping center virtually to see what shops are available.
   - **Responding** The app asks whether the user wants to visit their favorite snack bar on the way to the chosen shop.

(*Continued*)

2. Navigation apps tend to be smartphone-based, so it is worth exploring other styles to see what insights they may bring. We had the following thoughts, but you may have had others. The navigation app needs to be mobile so that the user can move around to find the relevant shop. Using voice or gesture interfaces is one option, but this could still be delivered through a mobile device. Thinking more broadly, perhaps a haptic interface that guides the user to the required location might suffice. Smart interfaces, such as one built into the environment is an alternative, but privacy issues may arise if an individual's data is displayed for all to see. ■

12.3.2 Expanding the Initial Conceptual Model

The previous elements represent the core of the conceptual model. For prototyping or testing with users, these ideas need some expansion. Examples include which functions the product will perform and which the user will perform, how those functions are related, and what information is required to support them. Some of this will have been considered during the requirements activity and will evolve after prototyping and evaluation.

What Functions Will the Product Perform?

This question is about whether the product or the user takes responsibility for different parts of the overall goal. For example, the travel organizer is intended to suggest specific vacation options for a group of people, but is that all it should do? What about if it automatically reserved the bookings? Or does it wait until it is given a preferred choice? In the case of visa requirements, will the travel organizer simply provide the information, or link to visa services? Deciding what the system will do and the user will do is sometimes called *task allocation*. This trade-off has cognitive implications (see Chapter 4, "Cognitive Aspects") and affects social aspects of collaboration (see Chapter 5, "Social Interaction"). If the cognitive load is too high for the user, then the device may be too stressful to use. On the other hand, if the product has too much control and is too inflexible, then it may not be used at all.

Another decision is which functions to hard-wire into the product and which to leave under software control, thereby indirectly in the control of the human user.

How Are the Functions Related to Each Other?

Functions may be related temporally; for example, one must be performed before another, or two can be performed in parallel. They may also be related through any number of possible categorizations, for instance, all functions relating to privacy on a smartphone or all options for viewing photographs on a social networking site. The relationships between tasks may constrain use or may indicate suitable task structures within the product. For example, if one task depends on another, the order in which tasks can be completed may need to be restricted. If use cases or other detailed analysis of the tasks have been generated, these will help. Different styles of requirements (for example, stories or atomic requirements shell) provide different levels of detail, so some of this information will be available, and some will evolve as the design team explores and discusses the product.

What Information Is Needed?

What data is required to perform a task? How is this data to be transformed by the system? Data is one of the categories of requirements identified and captured through the requirements activity. During conceptual design, these requirements are considered to ensure that the model provides the information needed to perform the task. Detailed issues of structure and display, such as whether to use an analog display or a digital display, will more likely be dealt with during the concrete design activity, but implications arising from the type of data to be displayed may impact conceptual design issues.

For example, identifying potential vacations for a group of people using the travel organizer requires the following: what kind of vacation is required; available budget; preferred destinations (if any); preferred dates and duration (if any); how many people it is for; and are there any special requirements (such as disability) within the group? To perform the function, the system needs this information and must have access to detailed vacation and destination descriptions, booking availability, facilities, restrictions, and so on.

Initial conceptual models may be captured in *wireframes*—a set of documents that show structure, content, and controls. Wireframes may be constructed at varying levels of abstraction, and they may show part of the product or a complete overview. Chapter 13, "Interaction Design in Practice," includes more information and some examples.

12.4 Concrete Design

Conceptual design and concrete design are closely related. The difference between them is rather a matter of changing emphasis: during design, conceptual issues will sometimes be highlighted, and at other times, concrete detail will be stressed. Producing a prototype inevitably means making some concrete decisions, albeit tentatively, and since interaction design is iterative, some detailed issues will come up during conceptual design, and vice versa.

Designers need to balance the range of environmental, user, data, usability, and user experience requirements with functional requirements. These are sometimes in conflict. For example, the functionality of a wearable interactive product will be restricted by the activities the user wants to perform while wearing it; a computer game may need to be learnable but also challenging.

There are many aspects to the concrete design of interactive products: visual appearance such as color and graphics, icon design, button design, interface layout, choice of interaction devices, and so on. Chapter 7 introduces several interface types, together with their associated design considerations, guidelines, principles, and rules, which help designers ensure that their products meet usability and user experience goals. These represent the kinds of decision that are made during *concrete design*.

Concrete design also deals with issues related to user characteristics and context. Two aspects that have drawn particular attention for concrete design are discussed in this section: accessibility and inclusiveness; and designing for different cultures. Accessibility and inclusiveness were introduced in Section 1.6. *Accessibility* refers to the extent to which a product is accessible to as many people as possible, while *inclusiveness* means being fair, open, and equal to everyone. The aim of inclusive design is to empower users in their everyday and working lives (Rogers and Marsden, 2013).

A range of input and output modes is available for interaction design. Apart from standard keyboard, mouse, and touchscreen, there are also different pointing devices and

keyboards, screen readers, refreshable Braille, and eye-tracking, among others. Regardless of which alternate input or output modalities are used, interactive interfaces must be flexible enough to work with these various devices. This is particularly important for users with disabilities, who may be unable to use pointing devices or standard keyboards and may instead interact using a head or mouth stick, voice recognition, video with captions, audio transcripts, and so on.

Making an interface accessible involves engaging users with disabilities in the development process to better understand their needs and using the Web Content Accessibility Guidelines (WCAG), which applies to all interfaces, not just web-based interfaces (see Box 16.2). When interfaces are designed to be accessible, they not only work for people with disabilities, but they also provide flexibility to other users without disabilities who are faced with temporary or situational impairments, for example, a driver who is unable to look at a display screen or a train passenger watching a video without disturbing other passengers.

Interfaces that are not accessible can lead to various forms of discrimination. For instance, air fares are often lower if purchased through a website. If that website is inaccessible for blind consumers, for example, and those consumers have to use the call center, they may unknowingly be charged higher fares for the same flight (Lazar et al., 2010). Many companies use online job applications, but if the online site is inaccessible, job applicants may be forced to identify themselves as having a disability before applying for a job. Similarly, when job aggregator websites (with information about jobs at many different employers) are inaccessible and individuals who are blind are told to call on the phone as an accommodation, they frequently aren't told about many or even any of the jobs available (Lazar et al., 2015).

There are resources available to help design for inclusivity, accessibility, and flexibility, such as Microsoft's inclusive design toolkit.

Microsoft's inclusive design toolkit has some useful and interesting resources. You can find out more at **https://www.microsoft.com/design/inclusive**

Aspects of cross-cultural design include use of appropriate language(s), colors, icons and images, navigation, and information architecture (Rau et al., 2013). These are all important for concrete design; however, tensions between local cultures and HCI principles have been highlighted (Winschiers-Theophilus and Bidwell, 2013) together with a desire to reframe human-computer interaction (HCI) through local and indigenous perspectives (Abdelnour-Nocera et al., 2013). These concerns not only impact concrete design but also wider issues such as what to design and how to design in order to be accepted by the target user group. For example, Gary Marsden et al. (2008) warn of the problems in seeing a user's need and attempting to meet that need without first asking the community if they too recognize that need. For one approach on how to address this concern, see Case Study 12.2.

12.5 Generating Prototypes

This section illustrates how prototypes may be used in design, and it demonstrates how prototypes may be generated from the output of the requirements activity—producing a storyboard from a scenario and an index card-based prototype from a use case. Both of these are low-fidelity prototypes.

12.5.1 Generating Storyboards

A *storyboard* represents a sequence of actions or events that the user and the product go through to achieve a goal. A *scenario* is one story about how a product may be used to achieve that goal. A storyboard can be generated from a scenario by breaking the scenario into a series of steps that focus on interaction and creating one scene in the storyboard for each step. The purpose for doing this is twofold: first to produce a storyboard that can be used to get feedback from users and colleagues and second to prompt the design team to consider the scenario and the product's use in more detail. For example, consider the scenario for the travel organizer developed in Chapter 11. This can be broken down into six main steps.

1. Will, Sky, and Eamonn gather around the organizer, but Claire is at her mother's house.
2. Will tells the organizer their initial idea of a sailing trip in the Mediterranean.
3. The system's initial suggestion is that they consider a flotilla trip, but Sky and Eamonn aren't happy.
4. The travel organizer shows them some descriptions written by young people about flotilla trips.
5. Will confirms this recommendation and asks for details.
6. The travel organizer emails details of the different options.

Notice that the first step sets the context, and later steps focus more on the goal. Breaking the scenario into steps can be achieved in different ways. The purpose of working from the scenario is for the design team to think through the product and its use, and so the steps are not as important as the thinking that happens through the process. Also, notice that some of these events are focused solely on the travel organizer's interface, and some are concerned with the environment. For example, the first one talks about the family gathering around the organizer, while the fourth and sixth are focused on the travel organizer. Storyboards can focus on the screens or on the environment, or a mixture of both. Either way, sketching out the storyboard will prompt the design team to think about design issues.

For example, the scenario says nothing about the kinds of input and output devices that the system might use, but drawing the organizer forces the designer to think about these things. There is some information in the scenario about the environment within which the system will operate, but drawing the scene requires specifics about where the organizer will be located and how interaction will continue. When focusing on the screens, the designer is prompted to consider issues including what information needs to be available and what information needs to be output. This all helps to explore design decisions and alternatives, but it is also made more explicit because of the drawing act.

The storyboard in Figure 12.12 includes elements of the environment and some of the screens. While drawing this, various questions came to mind such as how can the interaction be designed for all of the family? Will they sit or stand? How to handle remote participants?

What kind of help needs to be available? What physical components does the travel organizer need? How to enable all of the family to interact with the system (notice that the first scene uses voice input while other scenes have a keyboard option as well)? and so on. In this exercise, the questions it prompts are just as important as the end product.

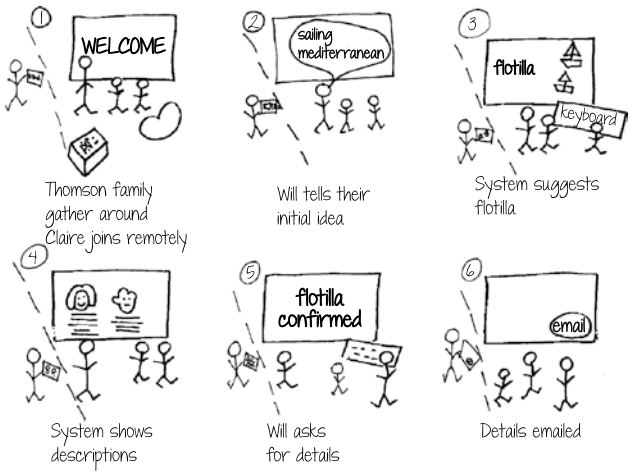


Figure 12.12 The storyboard for the travel organizer

ACTIVITY 12.4

Activity 11.4 in Chapter 11 developed a futuristic scenario for the one-stop car shop. Using this scenario, develop a storyboard that focuses on the environment of the user. As you draw this storyboard, write down the design issues that it prompts.

Comment

The following is based on the scenario in the comment for Activity 11.4. This scenario breaks down into five main steps.

1. The user arrives at the one-stop car shop.
2. The user is directed into an empty booth.
3. The user sits down in the racing car seat, and the display comes alive.
4. The user can view reports.
5. The user can take a virtual reality drive in their chosen car.

The storyboard is shown in Figure 12.13. Issues that arose while drawing this storyboard included how to display the reports, what kind of virtual reality equipment is needed, what input devices are needed—a keyboard or touchscreen, a steering wheel, clutch, accelerator, and brake pedals? How much like actual car controls do the input devices need to be? You may have thought of other issues.

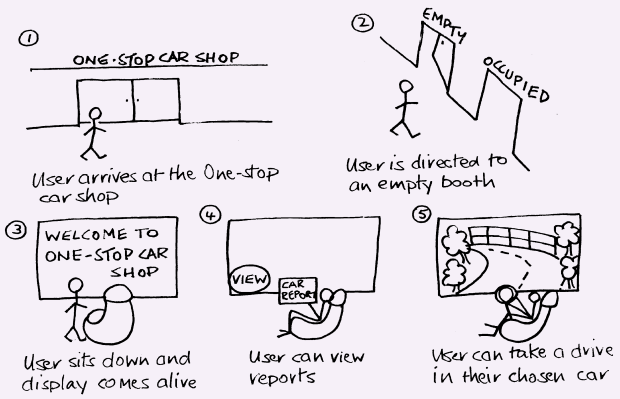


Figure 12.13 The storyboard generated from the one-stop car shop scenario in Activity 11.4 ■

12.5.2 Generating Card-Based Prototypes

Card-based prototypes are commonly used to capture and explore elements of an interaction, such as dialog exchanges between the user and the product. The value of this kind of prototype lies in the fact that the interaction elements can be manipulated and moved around in order to simulate interaction with a user or to explore the user's end-to-end experience. This may be done as part of the evaluation or in conversations within the design team. If the storyboard focuses on the screens, this can be translated almost directly into a card-based prototype and used in this way. Another way to produce a card-based prototype is to generate one from a use case output from the requirements activity.

For example, consider the use cases for the visa requirements aspect of the group travel organizer presented in Section 11.6. The first, less-detailed use case provides an overview of the interaction, while the second one is more detailed.

This second use case can be translated into cards as follows. For each step in the use case, the travel organizer will need to have an interaction component to deal with it, for example, input via a button, menu option, or voice, and output via a display or sound. By stepping through the use case, a card-based prototype can be developed that covers the required behavior, and different designs can be considered. For example, Figure 12.14 shows six dialog elements on six separate cards. The set on the left has been written in friendlier language, while the set on the right is more official. These cover steps 1, 2, 3, 4, and 5.

The alternative courses, for example those dealing with error messages, would also each have a card, and the tone and information contained in the error message could be evaluated with users. For example, step 7.1 might translate into a simple "No visa information is available," or a more helpful, "I am not able to find visa information for you to visit your chosen destination. Please contact the <*destination country*>'s embassy."

These cards can be shown to potential users of the system or fellow designers to get informal feedback. In this case, we showed these cards to a colleague, and through discussion of the application and the cards, concluded that although the cards represent one interpretation

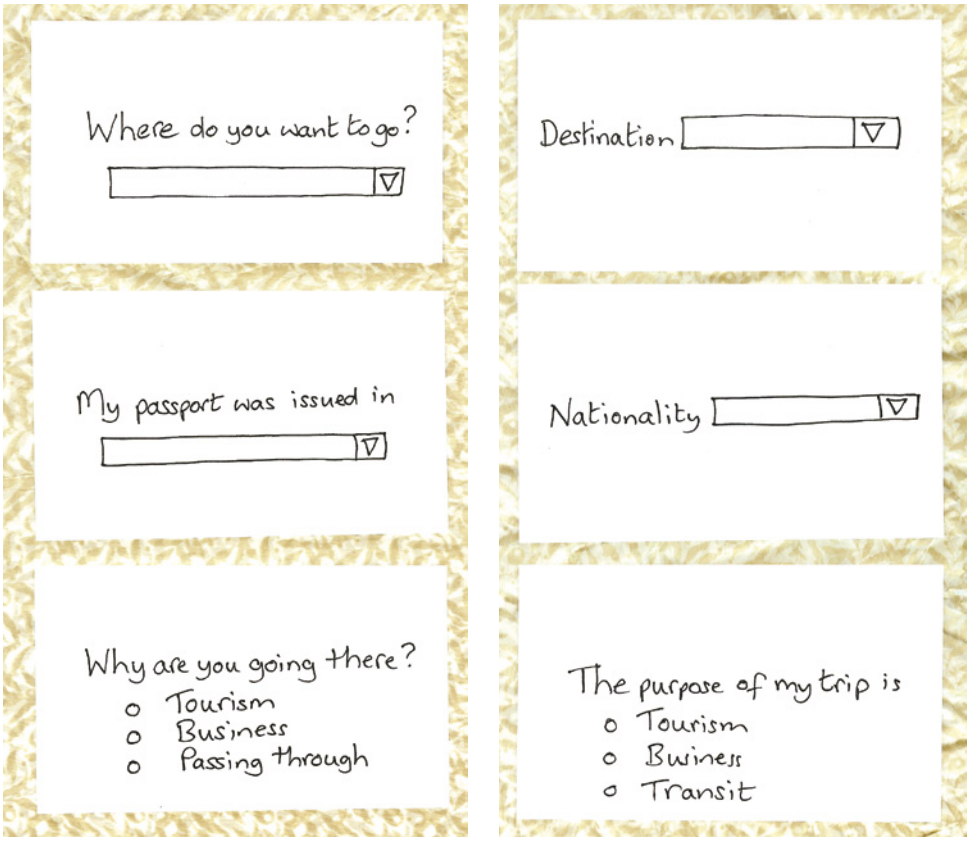


Figure 12.14 Cards 1–3 of a card-based prototype for the travel organizer

of the use case, they focus too much on an interaction model that assumes a WIMP/GUI interface. Our discussion was informed by several things including the storyboard and the scenario. One alternative would be to have a map of the world, and users can indicate their destination and nationality by choosing one of the countries on the map; another might be based around national flags. These alternatives could be prototyped using cards and further feedback obtained. Cards can also be used to elaborate other aspects of the concrete design, such as icons and other interface elements.

ACTIVITY 12.5

Look at the storyboard in Figure 12.4. This storyboard shows Christina exploring the Acropolis in search of information about the pottery trade. In the second scene in the top row, Christina "adjusts the preferences to find information about the pottery trade in Ancient Greece." Many interaction icons have become standardized, but there isn't a standard one for "pottery trade." Suggest two alternative icons to represent this and draw them on separate

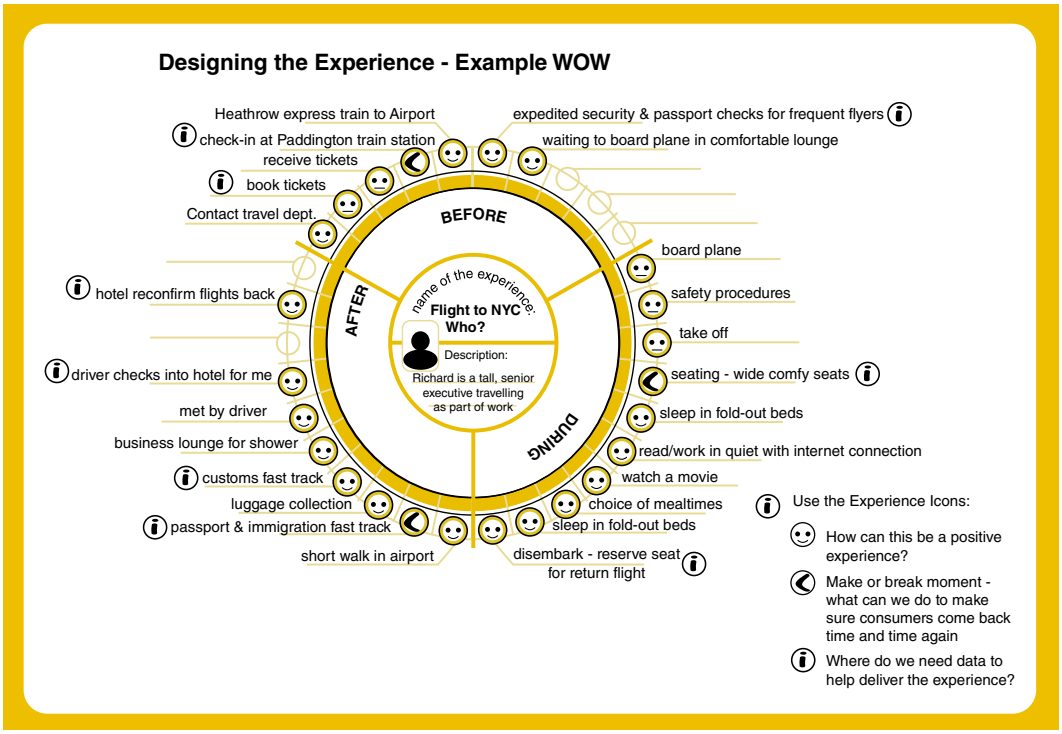
cards. Using the storyboard in Figure 12.4 and the two cards, try out the different icons with a friend or colleague to see what they understand by your two icons.

Comment

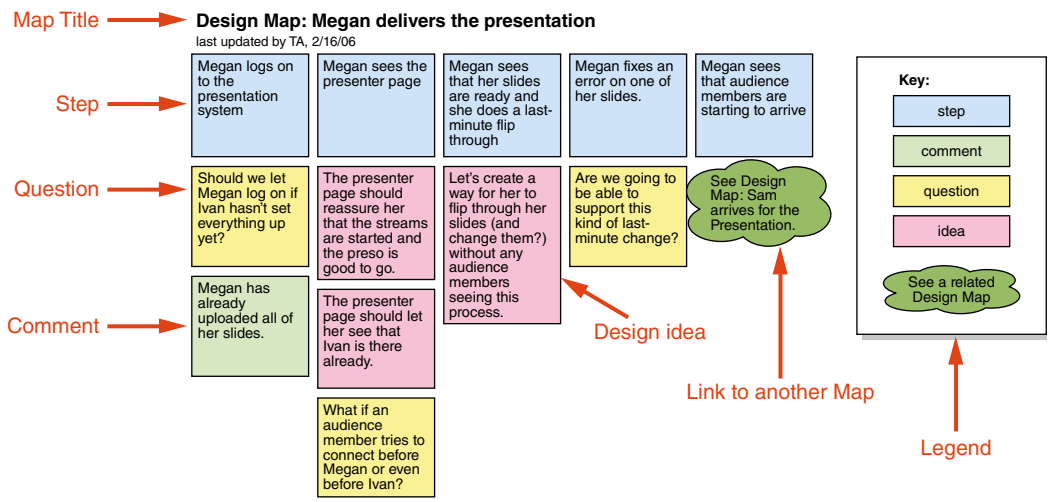
The two cards we drew are shown in Figure 12.15. The first is simply an Ancient Greek pot, while the second attempts to capture the idea of a pottery seller in the market. When we stepped through the storyboard with a colleague and showed them these alternatives, both were found to require improvement. The pot on its own did not capture the pottery trade, and it wasn't clear what the market seller represented, but there was a preference for the latter, and the users' feedback was useful.

Figure 12.15 Two icons to represent "pottery trade" for the new mobile device for exploring historic sites depicted in the storyboard of Figure 12.4 ■

A set of card-based prototypes that cover a scenario from beginning to end may be the basis of a more detailed prototype, such as an interface or screen sketch, or it may be used in conjunction with personas to explore the user's end-to-end experience. This latter purpose is also achieved by creating a visual representation of the user's experience. These representations are variably called a *design map* (Adlin and Pruitt, 2010), a *customer journey map*



(a)



(b)

Figure 12.16 (a) An experience map using a wheel representation and (b) an example timeline design map illustrating how to capture different issues

Source: (a) LEGO (b) Adlin and Pruitt (2010), p. 134. Used courtesy of Morgan Kaufmann

(Ratcliffe and McNeill, 2012), or an *experience map*. They illustrate a user's path or journey through the product or service and are usually created for a particular persona and based on a particular scenario, hence giving the journey sufficient context and detail to bring the discussions to life. They support designers in considering the user's overall experience when achieving a particular goal and are used to explore and question the designed experience and to identify issues that have not been considered so far. They may be used to analyze existing products and to collate design issues, or as part of the design process.

There are many different types of representation of varying complexities. Two main ones are: the wheel and the timeline. The wheel representation is used when an interaction phase is more important than an interaction point, such as for a flight (see Figure 12.16(a) for an example). The timeline is used where a service is being provided that has a recognizable beginning and end point, such as purchasing an item through a website (an example of a timeline representation is shown in Figure 11.7(b)—look for the smiley faces). Figure 12.16(b) illustrates the structure of a timeline and how different kinds of issues may be captured, such as, questions, comments, and ideas.

To generate one of these representations, take one persona and two or three scenarios. Draw a timeline for the scenario and identify the interaction points for the user. Then use this as a discussion tool with colleagues to identify any issues or questions that may arise. Some people consider the user's mood and identify pain points, sometimes the focus will be on technical issues, and at other times this can be used to identify missing functionality or areas of under-designed interaction.

This video illustrates the benefits of experience mapping using a timeline: **http://youtu.be/eLT_Q8sRpyI**.

To read about the main elements of a customer journey map, when you need them, and how to construct them, see this article: **https://www.nngroup.com/articles/customer-journey-mapping/**.

BOX 12.3

Involving Users in Design: Participatory Design

Participatory design (PD) emerged in Scandinavia in the late 1960s and early 1970s. There were two influences on this early work: the desire to be able to communicate information about complex systems and the labor union movement pushing for workers to have democratic control over changes in their work. In the 1970s, new laws gave workers the right to have a say in how their working environment was changed, and such laws are still in force today.

(*Continued*)

The idea that those who use information technology will play a critical role in its design, and in particular that they will engage in active and genuine participation with the design itself, is still central to participatory design (Simonsen and Robertson, 2012). But the approach has evolved considerably in response to political, social, and technological changes (Bannon et al., 2018). In addition, many approaches to technology design include participation with users, so what makes participatory design different?

In a review of research in the PD conference 2002–2012, Kim Halskov et al. (2015) wanted to understand the different definitions of 'participation' as evidenced by those papers. They identified three approaches.

- Implicit, meaning that the paper wasn't clear about the details of participation
- Users as full participants in the design process, which goes beyond simple involvement of users, but extends to understanding the user's point of view, and regarding what users know as being important
- Mutual learning between users and designer

One of the key questions for participatory design today is how to handle scale: How to ensure participation by a community when that community includes several hundreds or thousands of people? How to ensure participation when the data collected can be from many different sources and without explicit agreement? How to ensure participation when users extend over several countries?

Daniel Gooch et al. (2018) designed an approach to facilitate citizen engagement in a smart city project. They used an integrated approach of online and offline activities that was tailored to local contexts and showed how it is possible to engage citizens in a way that addresses citizens' current concerns. They also identified four key challenges to utilizing participatory design on an urban scale.

- Balancing scale with the personal. In particular, the need to engage face-to-face with potential participants.
- Who has control of the process? If participants are to have a meaningful say in what is designed, they need to have some of the power, but sometimes regulations mitigate against this.
- Who is participating? In a city, there are many diverse stakeholders, yet it is important to include all sections of the society to avoid bias.
- Integrating citizen-led work with local authorities. Regulations set by local authorities can be an obstacle to innovation within a city context.

Case Study 12.2 describes an extension to participatory design, called *community-based design*, developed for the local situation in South Africa. ■

CASE STUDY 12.2

Deaf Telephony

This case study by Edwin Blake, William Tucker, Meryl Glaser, and Adinda Freudenthal discusses their experiences of community-based design in South Africa. The process of community-based co-design is one that explores various solution configurations in a multidimensional design space whose axes are the different dimensions of requirements and the various dimensions of designer skills and technological capabilities. The bits of this space that one can "see" are determined by one's knowledge of the user needs and one's own skills. Co-design is a way of exploring that space in a way that alleviates the myopia of one's own viewpoint and bias. As this space is traversed, a trajectory is traced according to one's skills and learning and according to the users' expressed requirements and their learning.

The project team set out to assist South African deaf people to communicate with each other, with hearing people, and with public services. The team has been working for many years with a deaf community that has been disadvantaged due both to poverty and hearing impairment. The story of this wide-ranging design has been one of continual fertile (and on occasion frustrating) co-design with this community. The team's long-term involvement has meant that they have transformed aspects of the community and that they have themselves been changed in what they view as important and in how they approach design. Figure 12.17 illustrates one participant's view of communication, captured during a community engagement event, and Figure 12.18 shows two participants discussing design using sign language.

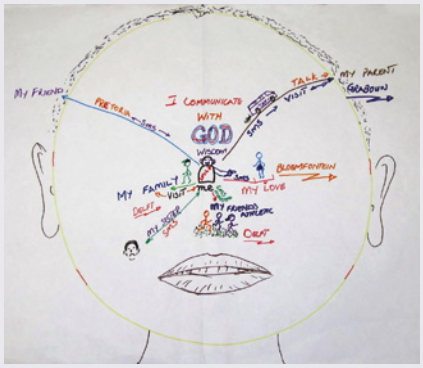


Figure 12.17 One participant's view of communication
Source: Edwin Blake

(*Continued*)

Figure 12.18 Participants discussing design in sign language
Source: Helen Sharp

Deaf users in this community started out knowing essentially nothing about computers. Their first language is South African Sign Language (SASL), and this use of SASL is a proud sign of their identity as a people. Many are also illiterate or semi-literate. There are a large number of deaf people using SASL; in fact, there are more using it than with some of the smaller official languages. Since the advent of democracy in 1994, there has been an increasing empowerment of deaf people, and it is accepted as a distinct language in its own right.

In this case study on id-book.com, a brief historical overview of the project and the various prototypes that formed nodes in a design trajectory are presented. The methodology of Action Research and its cyclical approach to homing in on an effective implementation is reviewed. An important aspect of the method is how it facilitates learning by both the researchers and the user community so that together they can form an effective design team. Lastly, such a long-term intimate involvement with a community raises important ethical issues, which are fundamentally concerns of reciprocity. ■

ACTIVITY 12.6

Design thinking has been described as an approach to problem-solving and innovative design that focuses on understanding what people want and what technology can deliver. It is derived from professional design practice, and it is often viewed as having five stages that together evolve a solution: empathize, define, ideate, prototype, and test. A slightly different view of design thinking, according to IDEO (https://www.ideou.com/pages/design-thinking), emphasizes human needs, empathy, and collaboration by looking at the situation through three lenses: desirability, feasibility, and viability.

Design thinking has become very popular, but some have questioned its benefits and implications. This activity invites you to decide for yourself.

Click the following links, and do some investigation yourself around the idea of design thinking. Based on what you find, do you think the turn toward design thinking overall is beneficial or damaging to interaction design?

Jon Kolko's (2018) article:
http://interactions.acm.org/archive/view/may-june-2018/the-divisiveness-of-design-thinking
Natasha Jen's (2017) presentation:
https://vimeo.com/228126880
Dan Nessler's (2016) article:
https://medium.com/digital-experience-design/how-to-apply-a-design-thinking-hcd-ux-or-any-creative-process-from-scratch-b8786efbf812

Comment

Design thinking is similar to the approaches espoused by user-centered design and the notion of design thinking has been embraced by many designers and organizations. Nevertheless, the way in which it has been popularized has resulted in some heavy criticism too. In her presentation, Natasha Jen criticizes the simple five-stage process and invites proponents to share the evidence of its success and its outcomes so that it can be improved.

Jon Kolko (2018) believes that this surge of interest in design thinking "will leave behind two benefits: validation of the design profession as real, intellectual, and valuable—and a very large need for designers who can make things." However, he also points out that it has been popularized at a simplistic level of detail.

At the end of the day, what this suggests is that design is a creative activity supported by techniques, tools, and processes, but it cannot be boiled down into a particular process or set of techniques—design involves "the habit of continually doing things in new ways in order to make a difference," as stated by Dan Nessler. ■

12.6 Construction

As prototyping and building alternatives progresses, development will focus more on putting together components and developing the final product. This may take the form of a physical product, such as a set of alarms, sensors, and lights, a piece of software, or both. Whatever the final form, it is unlikely that anything will need to be developed from scratch, as there are many useful (in some cases essential) resources to support development. Here we introduce two kinds of resources: physical computing kits and software development kits (SDKs).

12.6.1 Physical Computing

Physical computing is concerned with how to build and code prototypes and devices using electronics. Specifically, it is the activity of "creating physical artifacts and giving them behaviors through a combination of building with physical materials, computer programming, and circuit building" (Gubbels and Froehlich, 2014). Typically, it involves designing things, using a printed circuit board (PCB), sensors (for instance push buttons, accelerometers, infrared, or temperature sensors) to detect states, and output devices (such as displays, motors, or buzzers) that cause some effect. An example is a "friend or foe" cat detector that senses, via an accelerometer, any cat (or anything else for that matter) that tries to push through a family's cat door. The movement triggers an actuator to take a photo of what came through the cat door using a webcam positioned on the back door. The photo is uploaded to a website that alerts the owner if the image does not match that of their own cat.

A number of physical computing toolkits have been developed for educational and prototyping purposes. One of the earliest is Arduino (see Banzi, 2009). The goal was to enable artists and designers to learn how to make and code physical prototypes using electronics in a couple of days, having attended a workshop. The toolkit is composed of two parts: the Arduino board (see Figure 12.19), which is the piece of hardware that is used to build objects, and the Arduino integrated development environment (IDE), which is a piece of software that makes it easy to program and upload a sketch (Arduino's name for a unit of code) to the board. A sketch, for example, might turn on an LED when a sensor detects a change in the light level. The Arduino board is a small circuit that contains a tiny chip (the microcontroller). It has two rows of small electrical "sockets" that let the user connect sensors and actuators to its input and output pins. Sketches are written in the IDE using a simple processing language and then translated into the C programming language and uploaded to the board.

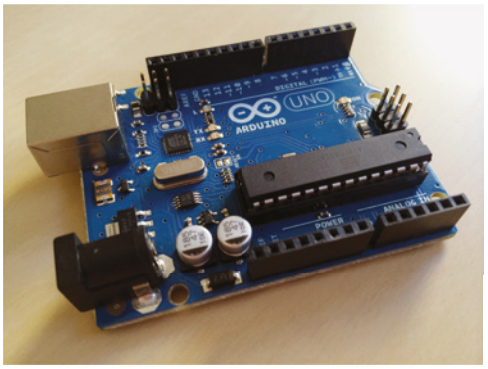


Figure 12.19 The Arduino board

Source: Used courtesy of Dr Nicolai Marquardt

There are other toolkits that have been developed, based on the basic Arduino kit. The most well-known is the LilyPad, which was co-developed by Leah Beuchley (see Figure 12.20 and her interview at the end of Chapter 7). It is a set of sewable electronic components for building fashionable clothing and other textiles. The Engduino is a teaching tool based on the Arduino LilyPad; it has 16 multicolor LEDs and a button, which can be used to provide

visual feedback and simple user input. It also has a thermistor (that senses temperature), a 3D accelerometer (that measures accelerations), and an infrared transmitter/receiver that can be used to transmit messages from one Engduino to another.

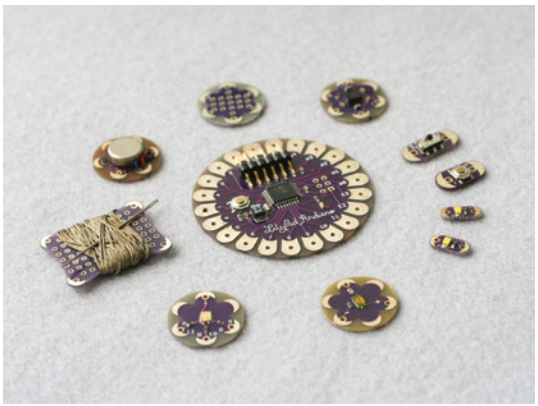


Figure 12.20 The Lilypad Arduino kit
Source: Used courtesy of Leah Beuchley

Watch this video that introduces Magic Cubes—a novel toolkit that is assembled from six sides that are slotted together to become an interactive cube that lights up in different colors, depending on how vigorously it is shaken. Intended to encourage children to learn, share, and fire their imagination to come up with new games and other uses, see it in action at **https://uclmagiccube.weebly.com/video.html**.

Other kinds of easy-to-use and quick-to-get-started physical toolkits, intended to provide new opportunities for people to be inventive and creative, are Senseboard (Richards and Woodthorpe, 2009), Raspberry Pi (https://www.raspberrypi.org/), .NET Gadgeteer (Villar et al., 2012), and MaKey MaKey (Silver and Rosenbaum, 2012). The MaKey MaKey toolkit is composed of a printed circuit board with an Arduino microcontroller, alligator clips, and a USB cable (see Figure 12.21). It communicates with a computer to send key presses, mouse clicks, and mouse movements. There are six inputs (the four arrow keys, the space bar, and a mouse click) positioned on the front of the board onto which alligator clips are clipped in order to connect with a computer via the USB cable. The other ends of the clips can be attached to any noninsulating object, such as a vegetable or piece of fruit. Thus, instead of using the computer keyboard buttons to interact with the computer, external objects such as bananas are used. The computer thinks MaKey MaKey is just like a keyboard or mouse. An example is to play a digital piano app using bananas as keys rather than keys on the computer keyboard. When they are touched, they make a connection to the board and MaKey MaKey sends the computer a keyboard message.

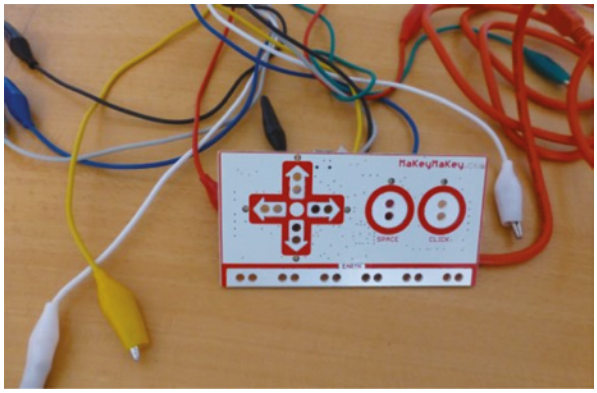


Figure 12.21 The MaKey MaKey toolkit
Source: Helen Sharp

One of the most recent physical computing systems is the BBC micro:bit (https://microbit .org, see Figure 12.22). Like Arduino, the micro:bit system consists of a physical computing device that is used in conjunction with an IDE. However, unlike Arduino, the micro:bit device contains a number of built-in sensors and a small display so that it is possible to create simple physical computing systems without attaching any components or wires. If desired, external components can still be added, but rather than the small electrical sockets of the Arduino, the micro:bit has an "edge connector" for this purpose. This is formed from a row of connection points that run along one edge of the device and allow it to be "plugged into" a range of accessories including larger displays, Xbox-style game controllers, and small robots. The micro:bit IDE, which runs in a web browser with no installation or setup process, supports a graphical programming experience based on visual "blocks" of code alongside text-based editing using a variant of JavaScript. This means that the micro:bit provides a great experience for young students and other beginner programmers, while also supporting more sophisticated programming. As a result, micro:bit has been widely adopted in schools around the world.

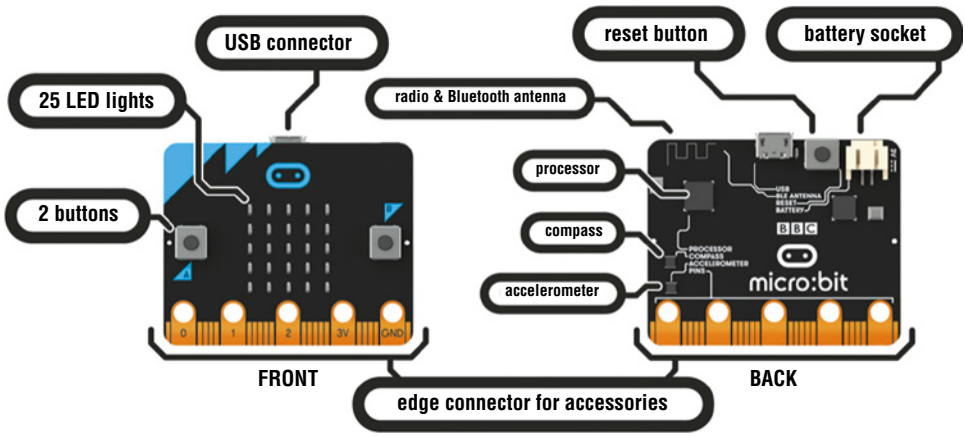


Figure 12.22 The BBC micro:bit
Source: https://microbit.org/guide/features. Used courtesy of Micro:bit Foundation

So far, physical toolkits have been aimed at children or designers to enable them to start programming through rapid creation of small electronic gadgets and digital tools (for example, Hodges et al., 2013, Sentance et al., 2017). However, Yvonne Rogers et al. (2014) demonstrated how retired people were equally able to be creative using the kit, turning "everyday objects into touchpads." They ran a series of workshops where small groups of retired friends, aged between their early 60s and late 80s, assembled and played with the MaKey MaKey toolkit (see Figure 12.23). After playing music using fruit and vegetables as input, they saw many new possibilities for innovative design. Making and playing together, however childlike it might seem at first, can be a catalyst for imagining, free thinking, and exploring. People are sometimes cautious to volunteer their ideas, fearing that they are easily squashed, but in a positive environment they can flourish. The right kind of shared experience can create a positive and relaxed atmosphere in which people from all walks of life can freely bounce ideas off each other.

Figure 12.23 A group of retired friends playing with a MaKey MaKey toolkit
Source: Helen Sharp

BOX 12.4

The Rise of the Maker Movement

The maker movement emerged in the mid-2000s. Following in the footsteps of the personal computer revolution and the Internet, some viewed it as the next big transformation that would modernize manufacturing and production (Hatch, 2014). Whereas the explosion of the Web was all about what it could do for us virtually, with a proliferation of apps, social media, and services, the maker movement is transforming how we make, buy, consume, and recycle physical things, from houses to clothes and food to bicycles. At its core is DIY—crafting physical things using a diversity of machines, tools, and methods collaboratively in workshops and makerspaces. In a nutshell, it is about inventing the future through connecting technologies, the Internet, and physical things.

(*Continued*)

While there have always been hobbyists tinkering away making radios, clocks, and other devices, the world of DIY making has been opened up to many more people. Affordable, powerful, and easy-to-use tools, coupled with a renewed focus on locally-sourced products and community-based activities, and a desire for sustainable, authentic, and ethically-produced products, has led to a groundswell of interest in "making." Fablabs (fabrication laboratories) first started appearing in cities throughout the world, offering a large physical space containing electronics and manufacturing equipment, including 3D printers, CNC milling machines, and laser cutters. Individuals bring their digital files to print and make things such as large 3D models, furniture, and installations—something that would have been impossible for them to do previously. Then smaller makerspaces started appearing in the thousands across the world, from Shanghai to rural India, again sharing production facilities for all to use and make. While some are small, for example sharing the use of a 3D printer, others are much larger and well resourced, offering an array of manufacturing machines, tools, and workspaces to make in.

Another development has been to build and program e-textiles using sewing machines and electronic thread. E-textiles comprise fabrics that are embedded with electronics, such as sensors, LEDs, and motors that are stitched together using conductive thread and conductive fabrics (Buechley and Qiu, 2014). An early example is the turn-signal biking jacket (developed by Leah Buechley and illustrated in Figure 1.4). Other e-textiles include interactive soft toys, wallpaper that sings when touched, and fashion clothing that reacts to the environment or events.

A central part of the maker movement involves tinkering (as discussed in section 12.2.4) and the sharing of knowledge, skills, know-how, and what you have made. The Instructables .com website is for anyone to explore, document, and share their DIY creations. Go to the Instructables site and take a look at a few of the projects that have been uploaded by makers. How many of them are a combination of electronics, physical materials, and pure invention? Are they fun, useful, or gadgety? How are they presented? Do they inspire you to make?

Another site, Etsy.com, is an online marketplace for people who make things to sell their crafts and other handmade items, which has grown in popularity over the past few years. It is designed to be easy for makers to use and to set up their store to sell to family, friends, and strangers across the world. Unlike corporate online sites, such as Amazon or eBay, Etsy is a place for craft makers to reach out to others and to show off their wares in ways that they feel best fit what they have made. This transition from "making" to "manufacturing," albeit on the limited scale of craft production, is an interesting phenomenon. Some authors believe that the trend will continue and that increasingly new products and new businesses will emerge from activities rooted in maker culture (Hodges et al., 2014).

In essence, the maker movement is about taking the DIY movement online to make it public, and in doing so, massively increase who can take part and how it is shared (Anderson, 2013). In his interview at the end of this chapter, Jon Froehlich explains more about the maker movement. ■

12.6.2 SDKs: Software Development Kits

A *software development kit* (SDK) is a package of programming tools and components that supports the development of applications for a specific platform, for example, for iOS on iPhone and iPad and for Android on mobile phone and tablet apps. Typically, an SDK includes an integrated development environment, documentation, drivers, and sample programming code to illustrate how to use the SDK components. Some also include icons and buttons that can easily be incorporated into the design. While it is possible to develop applications without using an SDK, it is much easier using such a powerful resource and so much more can be achieved.

For example, the availability of Microsoft's Kinect SDK has made the device's powerful gesture recognition and body motion tracking capabilities accessible. This has led to the exploration of many applications including elderly care and stroke rehabilitation (Webster and Celik, 2014), motion tracking in immersive games (Manuel et al., 2012), user identification using body lengths (Hayashi et al., 2014), robot control (Wang et al., 2013), and virtual reality (Liu et al, 2018).

An SDK will include a set of application programming interfaces (APIs) that allows control of the components without the developer needing to know the intricacies of how they work. In some cases, access to the API alone is sufficient to allow significant work to be undertaken, for instance, Eiji Hayashi et al. (2014) only needed access to the APIs. The difference between APIs and SDKs is explained in Box 12.5.

See the following websites to learn about two different types of SDKs and their use:

- Building voice-based services with Amazon's Alexa Skills Kit: **https://developer.amazon.com/alexa-skills-kit**.
- Constructing augmented reality experiences with Apple's ARKit: **https://developer.apple.com/arkit/**.

BOX 12.5

APIs and SDKs

SDKs consist of a set of programming tools and components, while an API is the set of inputs and outputs, that is, the technical interface to those components. To explain this further, an API allows different-shaped building blocks of a child's puzzle to be joined together, while an SDK provides a workshop where all of the development tools are available to create whatever size and shape blocks you desire, rather than using preshaped building blocks. An API therefore allows the use of pre-existing building blocks, while an SDK removes this restriction and allows new blocks to be created or even to build something without blocks at all. An SDK for any platform will include all of the relevant APIs, but it adds programming tools, documentation, and other development support as well. ■

In-Depth Activity

This in-depth activity builds upon the requirements activities related to the booking facility introduced at the end of Chapter 11.

1. Based on the information gleaned from the activity in Chapter 11, suggest three different conceptual models for this system. Consider each of the aspects of a conceptual model discussed in this chapter: interface metaphor, interaction type, interface type, activities it will support, functions, relationships between functions, and information requirements. Of these conceptual models, decide which one seems most appropriate and articulate the reasons why.
2. Using the scenarios generated for the online booking facility, produce a storyboard for the task of booking a ticket for one of the conceptual models in step 1. Show it to two or three potential users and record some informal feedback.
3. Considering the product's concrete design, sketch out the application's initial interface. Consider the design issues introduced in Chapter 7 for the chosen interface type. Write one or two sentences explaining your choices and consider whether the choice is a usability consideration or a user experience consideration.
4. Sketch out an experience map for the product. Use the scenarios and personas you generated previously to explore the user's experience. In particular, identify any new interaction issues that had not been considered previously, and suggest what could be done to address them.
5. How does the product differ from applications that typically might emerge from the maker movement? Do software development kits have a role? If so, what is that role? If not, why not?

Summary

This chapter explored the activities of design, prototyping, and construction. Prototyping and scenarios are used throughout the design process to test ideas for feasibility and user acceptance. We have looked at different forms of prototyping, and the activities have encouraged you to think about and apply prototyping techniques in the design process.

Key points

- Prototyping may be low fidelity (such as paper-based) or high fidelity (such as software-based).
- High-fidelity prototypes may be vertical or horizontal.
- Low-fidelity prototypes are quick and easy to produce and modify, and they are used in the early stages of design.

- Ready-made software and hardware components support the creation of prototypes.
- There are two aspects to the design activity: conceptual design and concrete design.
- Conceptual design develops an outline of what people can do with a product and what concepts are needed to understand how to interact with it, while concrete design specifies the details of the design such as layout and navigation.
- We have explored three approaches to help you develop an initial conceptual model: interface metaphors, interaction styles, and interface styles.
- An initial conceptual model may be expanded by considering which functions the product will perform (and which the user will perform), how those functions are related, and what information is required to support them.
- Scenarios and prototypes can be used effectively in design to explore ideas.
- Physical computing kits and software development kits facilitate the transition from design to construction.

Further Reading

BANZI, M. and SHILOH, M. (2014) *Getting Started with Arduino* (3rd ed.). Maker Media Inc. This hands-on book provides an illustrated step-by-step guide to learning about Arduino with lots of ideas for projects to work on. It outlines what physical computing is in relation to interaction design and the basics of electricity, electronics, and prototyping using the Arduino hardware and software environment.

GREENBERG, S., CARPENDALE, S., MARQUARDT, N. and BUXTON, B. (2012) *Sketching User Experiences*. Morgan Kaufmann. This is a practical introduction to sketching. It explains why sketching is important, and it provides useful tips to get the reader into the habit of sketching. It is a companion book to Buxton, B. (2007) *Sketching User Experiences*. Morgan Kaufmann, San Francisco.

INTERACTIONS MAGAZINE (2018) *Designing AI*. ACM. This issue of the *Interactions* magazine is all about design and different aspects of it including sketching, human-centered design for children, collaborative art, design capabilities, and the special topic of designing for AI.

LAZAR, J., GOLDSTEIN, D., and TAYLOR, A. (2015). *Ensuring Digital Accessibility Through Process and Policy*. Waltham, MA: Elsevier/Morgan Kaufmann Publishers. This book is about accessibility, bringing together knowledge in technology, law, and research. It includes a range of standards, regulations, methods, and case studies.

INTERVIEW with Jon Froehlich

Jon Froehlich is an Associate Professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington (UW) where he directs the Makeability Lab (http://makeabilitylab.io/), a cross-disciplinary research group focused on applying computer science and HCI to high-value social domains such as environmental sustainability and STE(A)M education. He has published more than 50 peer-reviewed publications; 11 have been honored with awards, including Best Papers at ACM CHI and ASSETS and a 10-Year Impact Award at UbiComp. Jon is a father of two, and he is increasingly passionate about CS4All—both as an educator and a researcher.

Can you tell us a bit about your research, what you do, and why you do it?
The goal of my research is to develop interactive tools and techniques to address pressing global challenges in areas such as accessibility, STE(A)M education, and environmental sustainability. To succeed at this work, I collaborate across disciplines with a focus on identifying long-term, ambitious research problems such as mapping the accessibility of the physical world via crowdsourcing plus computer vision that can also provide immediate, practical utility. Typically, my research involves inventing or reappropriating methods to sense physical or behavioral phenomena, leveraging techniques in computer vision (CV) and machine learning (ML) to interpret and characterize this data, and then building and evaluating interactive software or hardware tools uniquely enabled by these approaches. My research process is iterative, consisting of formative studies, which then inform the design and implementation of prototypes, followed by a series of evaluations, first in the lab and then eventually deployment studies of refined prototypes in the field.

What is the maker movement, and why are you so enthusiastic about it?
The maker movement emerged in the mid-2000s as an informal collection of hobbyists, engineers, artists, coders, and

craftspeople dedicated to playful creation, self-learning, and material design. While the movement builds on longstanding hobbyist and do-it-yourself (DIY) culture—for example, in woodworking and electronics—the movement was galvanized and accelerated by a series of socio-technical developments, including new, low-cost computational fabrication tools like CNC mills and 3D printers, the emergence of inexpensive and easy-to-use microcontroller platforms like Arduino and Raspberry Pi, online marketplaces like Adafruit and Sparkfun that made it easy to find and purchase parts, and social networks like Instructables, YouTube, and Thingiverse, which provided a forum for novices and experts alike to share and critique ideas, tutorials, and creations.

My enthusiasm for the maker movement stems both from my intrinsic excitement as a technologist in observing the creativity and creations of "makers" as well as from my perspectives as an educator and mentor in wondering how we can borrow from and adapt elements of the movement into formal education. While the maker movement is a relatively new phenomenon, its historical roots in education and learning science stretch back to pioneering educational thinkers like Maria Montessori, Jean Piaget, Seymour Papert, Lev Vygotsky, and others, all who emphasize the importance of learning through creation and experimentation, the role of peer mentorship, and how sharing work and soliciting feedback shapes thinking. For example, Papert's Constructionism learning theory places a critical focus not just on learning through making but on the social nature of design—that is, that ideas are shaped by the knowledge of an audience and the feedback provided by others.

I have tried to inject this philosophy into my undergrad and graduate teaching. As one example, students in my Tangible Interactive Computing course explore the materiality of interactive computing via design prompts such as making a new input device for a computer using lo-fi materials like conductive clay and fabric, breaking and remaking an existing electronic technology to reformulate its physical interaction, and combining computer vision and video cameras to create whole-body, gestural input. Students share and critique each other's work but also design outwardly beyond the confines of the classroom by sharing their results and design processes publicly (under pseudonyms, if preferred) via videos on YouTube, step-by-step tutorials on Instructables.com, and on the course website. Student-written Instructables in Tangible Interactive Computing, for example, have won awards and acquired more than 300,000 views and 1,900 favorites.

What are the advantages and challenges of working with communities to design products?

Much of my research involves designing and evaluating technologies for users who have different abilities, perspectives, and/or experiences from me and my research group—for example, early elementary school learners, people who use wheelchairs, or people with visual impairments. Thus, a key facet of our research and design process is employing methods from participatory design (or "co-design"), an approach to design that attempts to actively involve and empower target users throughout the design process from ideation to lo-fi prototyping to

(*Continued*)

summative evaluation. For example, in the MakerWear project (Kazemitabaar et al., 2017)—a wearable construction kit for children—we worked with children to gather design ideas and solicit critical feedback, to test initial designs, and to help co-design toolkit behavior and the overall look and feel. Similarly, we also involved professional STEM educators to help us improve our designs and think about corresponding learning activities. Finally, we ran a series of pilot studies followed by workshops in afterschool programs and a children's museum to examine what and how children make with MakerWear, what challenges arise, and how their designs differ from creations made with other toolkits (for example, in robotics).

This human-centered, participatory design approach offers many advantages, including ensuring that we are addressing real user problems, helping ground our design decisions through use and feedback from target stakeholders, and empowering our users to have a real voice in shaping outcomes (from which our participants of all ages seem to gain satisfaction). There are trade-offs, however. Soliciting ideas from target users in an unstructured and unprincipled manner may lead to poorly defined outcomes and suboptimal designs. When working with children, we often follow Druin's Cooperative Inquiry methodology (Guha et al., 2013), which provides a set of techniques and guidelines for co-design with children that helps to channel and focus their creativity and ideas. A second challenge is in recruiting and supporting co-design sessions: this is a resource-intensive process that requires time and effort from both stakeholders and the research team. To mitigate this challenge, we often work on establishing and maintaining longitudinal relationships with community groups like local schools and museums. Finally, not all projects are amenable to these methods (such as when timelines are particularly aggressive).

Have you encountered any big surprises in your work?

The life of a researcher is full of surprises—one must get comfortable with ambiguity and ending a research journey at an unpredictable location. My most significant surprises, however, have come from people: from my students, from my mentors, and from my collaborators. My research methods and ideas have been profoundly influenced in unexpected ways by colleagues like Professor Tamara Clegg who made me rethink how we can personalize STEM learning through opportunities in everyday life (what she calls "scientizing" life) and Professor Allison Druin who introduced me to and immersed me in children-oriented participatory design methods. (I could hear the excited shouts and joyful exclamations of Kidsteam from my office, and I couldn't resist finding out more, which fundamentally changed how I did research in STEM education.) My students never cease to surprise me, from 3D-printing gears to fix an aerial drone to developing an interactive sandbox that traces human movement using electromechanically controlled marbles to designing an e-textile shirt that senses and visualizes the wearer's changing physiology via integrated anatomical models.

What are your hopes for the future?

As a graduate student, I recall being asked, "What are the biggest open questions in HCI, and how does your research work

toward addressing them?" I found this question both profoundly interesting and profoundly startling because it forced me to think about the most significant open areas in my field and to (somewhat uncomfortably) confront the relationship between this answer and my research. At the risk of sounding overly ambitious, I would like to adapt this question, which serves as a guiding principle for my research but that I also hope will inspire others: "What are the most significant societal challenges across the world? What role can computer science, HCI, and design play in addressing those challenges? And where does your research/work fit?" As computation pervades nearly every aspect of our lives, I believe it is our role as technologists, designers, and practitioners to ask these questions of ourselves and to think about the political, economic, environmental, and social implications of our work. As a professor and educator, I am hopeful. This larger world-view framing of CS seems to resonate with younger generations and, I hope, will soon become the norm. ■