CHAPTER 11
DESIGN, PROTOTYPING, AND CONSTRUCTION

Objectives

The main aims of this chapter are to:

- Describe prototyping and 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 physical computing kits and software development kits, and their role in construction.

11.1 Introduction

Design activities begin once some requirements have been established. The design emerges iteratively, through repeated design–evaluation–redesign cycles involving users. Broadly speaking, there are two types of design: conceptual and concrete. The former is concerned with developing a conceptual model that captures what the product will do and how it will behave, while the latter is concerned with details of the design such as menu structures, haptic feedback, physical widgets, and graphics. As design cycles become shorter, the distinction between these two becomes blurred, but they are worth distinguishing because each emphasizes a different set of design concerns.

For users to evaluate the design of an interactive product effectively, designers must 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 design progresses, they become more polished, compact, and robust so that they resemble the

final product.

Broadly speaking, the design process may start from two distinct situations: when starting from scratch or when modifying an existing product. Much of design comes from the latter, and it is tempting to think that additional features can be added, or existing ones tweaked, without extensive investigation, prototyping, or evaluation. Although prototyping and evaluation activities can be reduced if changes are not significant, they are still valuable and should not be skipped.

In Chapter 10, we discussed some ways to identify user needs and establish requirements. In this chapter, we look at the activities involved in progressing a set of requirements through the cycles of prototyping to construction. We begin by explaining the role and techniques of prototyping and then explain how prototypes may be used in the design process. We end with an exploration of physical computing and software development kits (SDKs) that provide a basis for construction.

11.2 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. Having established some requirements, the next step is to try out design ideas through prototyping and evaluation cycles.

11.2.1 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. When you hear the term prototype, you may imagine 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, an electronic 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 through 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 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. He 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 very simple (some might even say bizarre) prototype, but it served its purpose of simulating scenarios of use. Advances in 3D printer technologies, coupled with reducing prices, have increased their use in design. It is now possible to take a 3D model from a software package and print a prototype. Even soft toys and chocolate may be 'printed' in this way (see Figure 11.1).

(a)

(b)

(c)

Figure 11.1 (a) Color output from a 3D printer: all the gears and rods in this model were 'printed' in one pass from bottom to top, and when one gear is turned, the others turn too. (b) James Bond's Aston Martin in Skyfall was in fact a 3D-printed model (http://www.telegraph.co.uk/technology/news/9712435/The-names-Printing-3D-Printing.html). (c) A teddy bear 'printed' from a wireframe design http://www.disneyresearch.com/project/printed-teddy-bears/

Source: (a) The Computer Language Company, Inc., courtesy of Alan Freedman (b) Courtesy of voxeljet and Propshop Modelmakers Ltd (c) Courtesy of Scott Hudson, Human–Computer Interaction Institute, Carnegie Mellon University.

Video showing a teddy bear being 'printed' is available at http://www.disneyresearch.com/project/printed-teddy-bears/

11.2.2 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 Schön (1983) and 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 out 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 your prototype will influence the kind of prototype you build. So, for example, if you want to clarify how users might perform a set of tasks and whether your proposed design would support them in this, you might produce a paper-based mockup. Figure 11.2 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 work. 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. In the development of SITU, a smart food nutrition scale and tablet application, a range of prototypes and representations were used from initial idea to final product. These included screen sketches, paper and cardboard mockups, wireframes, and many post-its.

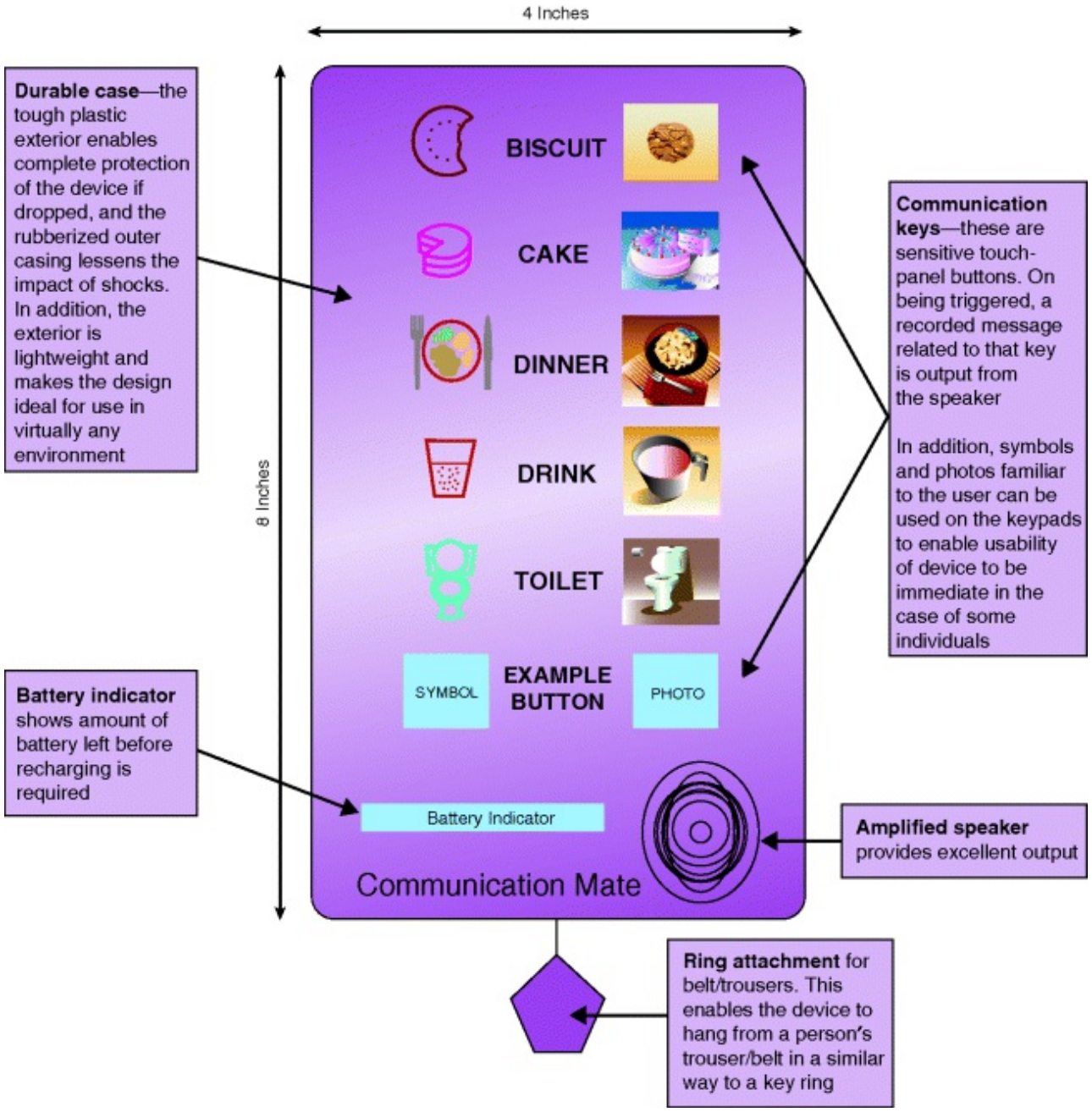

Figure 11.2 A paper-based prototype of a handheld device to support an autistic child

Source: Reprinted by permission of Sigil Khwaja.

Link to see the full story of SITU at
https://www.kickstarter.com/projects/situ/situ-smart-food-nutrition-scale

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 10.

11.2.3 Low-Fidelity Prototyping

A low-fidelity prototype does not look very much like the final product and does not 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 above 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 they support the exploration of alternative designs and ideas. This is particularly important in 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. They are for exploration only.

Storyboarding.

Storyboarding is one example of low-fidelity prototyping that is often used in conjunction with scenarios, as described in Chapter 10. 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 11.3 depicts a person (Christina) using a new mobile device for exploring historical 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.

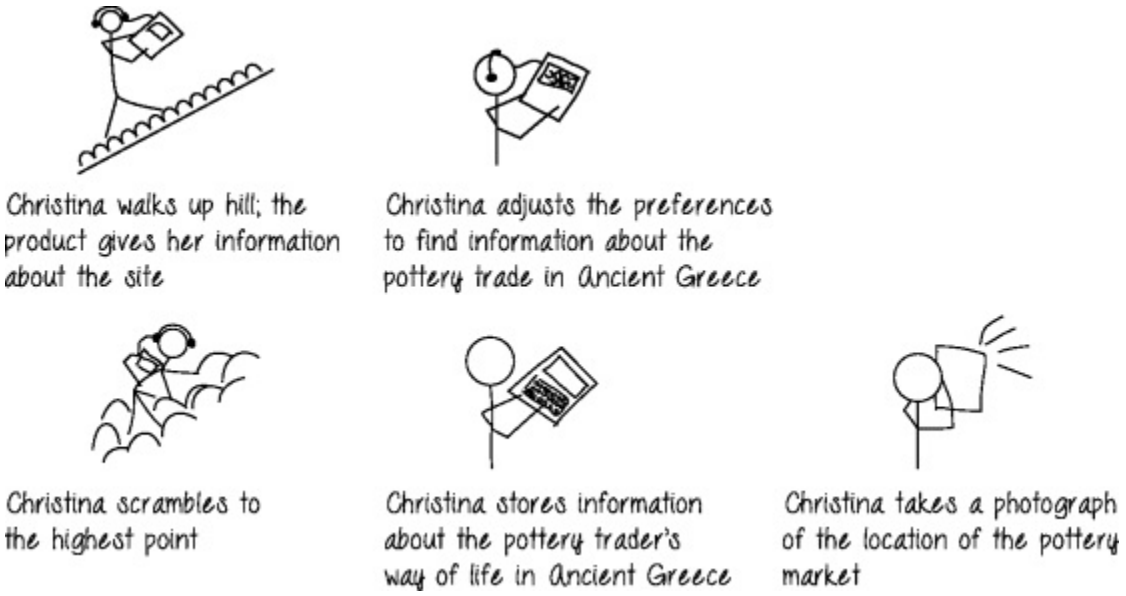

Figure 11.3 An example storyboard for a mobile device to explore ancient sites such as The Acropolis

Sketching.

Low-fidelity prototyping often relies on hand-drawn sketches, and many people find it difficult to engage in this activity because they are inhibited about the quality of their drawing, but as Greenberg et al (2012) put it, "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 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 you might require in a storyboard sketch, for example, include digital devices, people, emotions, tables, books, etc., 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 11.4. The next activity requires other sketching symbols, but they can still be drawn quite simply. Baskinger (2008) provides further tips for those new to sketching.

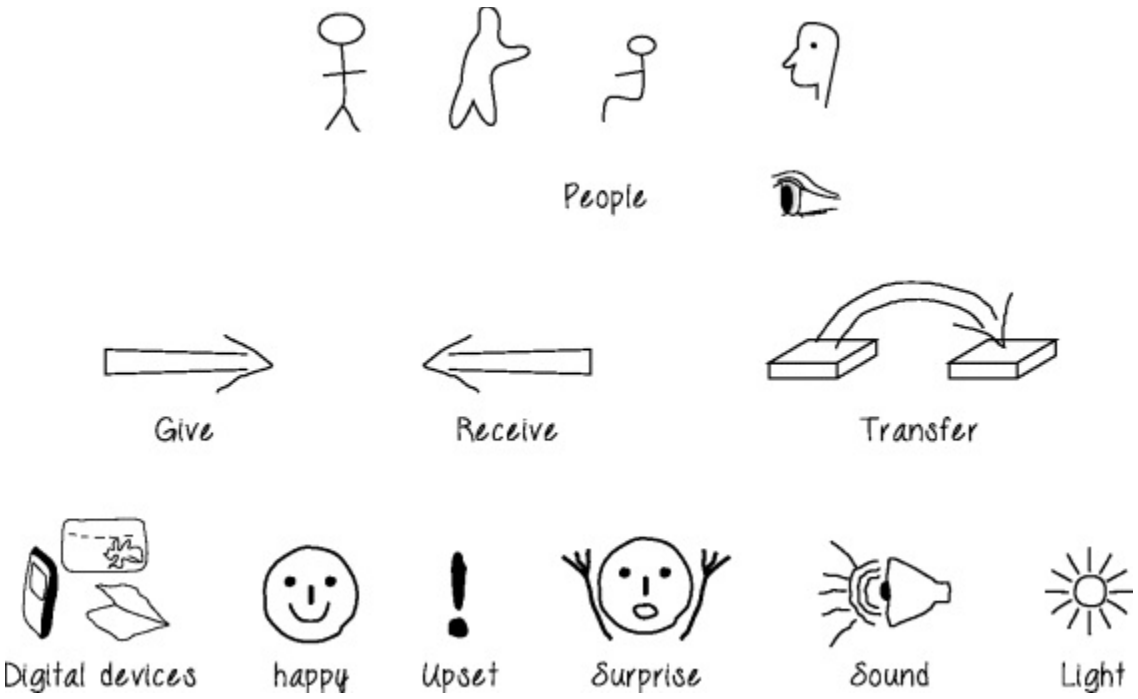

Figure 11.4 Some simple sketches for low-fidelity prototyping

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 is used often when developing websites. Each card represents a screen or one element of the interaction. 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 given in Section 11.6.2.

Activity 11.1

Produce a storyboard that depicts how to fill a car with gas (petrol).

Comment

Show/Hide

Wizard of Oz.

Another low-fidelity prototyping method called Wizard of Oz assumes that you have a software-based prototype. In 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 PinTrace, a robotic system that helps surgeons to position orthopedic pins accurately during the surgery of hip fractures (Molin, 2004), and to identify gestures for full body interaction with digital games (Norton et al, 2010).

11.2.4 High-Fidelity Prototyping

A high-fidelity prototype looks like the final product and/or provides more functionality than a low-fidelity prototype. For example, a prototype of a software system developed in Visual Basic is higher fidelity than a paper-based mockup; a molded piece of plastic with a dummy keyboard is a higher-fidelity prototype of the PalmPilot than the lump of wood. High-fidelity prototyping is useful for selling ideas to people and for testing out technical issues.

High-fidelity prototypes can be developed by modifying and integrating existing components – both hardware and software. 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). Banzi (2009) comments that: “Reusing existing technology is one of the best ways of tinkering. Getting cheap toys or old discarded equipment and hacking them to make them do something new is one of the best ways to get great results.” Bird et al (2009) describe how they used this approach to develop a tactile vision sensory substitution system, i.e. a system that translates a camera image of the user's environment into tactile stimulation on their body.

11.2.5 Compromises in Prototyping

By their very nature, prototypes involve compromises: the intention is to produce something quickly to test an aspect of the product. 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 11.1. 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 software-based prototyping, some of the compromises will still be fairly clear; for example, the response speed may

be slow, or the look and feel may not be finalised, or only a limited amount of functionality may be available.

BOX 11.1

The Anatomy of Prototyping: Filters and Manifestations

Lim et al (2008) propose a view of prototypes which focuses on their role as filters, i.e. to emphasize specific aspects of a product being explored by the prototype, and as manifestations of designs, i.e. 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.
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.

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 Tables 11.1 and 11.2. □

Table 11.1 Example variables of each filtering dimension

Filtering dimension	Example variables
Appearance	size; color; shape; margin; form; weight; texture; proportion; hardness; transparency; gradation; haptic; sound
Data	data size; data type (e.g. number; string; media); data use; privacy type; hierarchy; organization
Functionality	system function; users' functionality need
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 11.2 The definition and variables of each manifestation dimension

Manifestation dimension	Definition	Example variables
Material	Medium (either visible or invisible) used to form a prototype	Physical media, e.g. paper, wood, and plastic; tools for manipulating physical matters, e.g. knife, scissors, pen, and sandpaper; computational prototyping tools, e.g. Macromedia Flash and Visual Basic; physical computing tools, e.g. Phidgets and Basic Stamps; available existing artifacts, e.g. a beeper to simulate a heart attack
Resolution	Level of detail or sophistication of what is manifested (corresponding to fidelity)	Accuracy of performance, e.g. 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, e.g. 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

Two common compromises that often must be traded against each other are breadth of functionality provided 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).

Other compromises won't be obvious to a user of the system. For example, the internal structure of the product may not have been carefully designed, and the prototype may contain spaghetti code or be badly partitioned. One of the dangers of producing functional prototypes, i.e. ones that users can interact with automatically, is that the prototype can appear to be the final

product. Another is that developers may consider fewer alternatives because the prototype works and users like it. However, the compromises made in order to produce the prototype must not be ignored, particularly those that are less obvious from the outside. For a good-quality product, good engineering principles must be adhered to.

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

BOX 11.2

When to use high fidelity and when to use low fidelity prototypes

Table 11.3 summarizes proclaimed advantages and disadvantages of high- and low-fidelity prototyping. Component kits and pattern libraries for interface components (see Section 11.7 and Chapter 12) 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 11.1). Both high- and low-fidelity prototypes provide useful feedback during evaluation and design iterations. For example, Dhillon et al (2011) found that a low-fidelity video prototype elicited comparable user feedback as a high-fidelity one, but was quicker and cheaper to produce. In the context of usability testing, most studies have found that there is no difference between the low- and high-fidelity approach in terms of user feedback (Sauer and Sonderegger, 2009), although they present some

evidence that medium-fidelity prototypes are viewed as being less attractive than high- or low-fidelity ones. When exploring issues of content and structure, low-fidelity prototyping is preferable simply on the basis of cost, with the caveat that designers must be careful not to design technically infeasible capabilities on paper (Holmquist, 2005). The overriding consideration is the purpose of the prototype, and what level of fidelity is needed in order to get useful feedback. □

Table 11.3 Advantages and disadvantages of low- and high-fidelity prototypes

Type	Advantages	Disadvantages
Low-fidelity prototype	Lower development cost Evaluates multiple design concepts Useful communication device Addresses screen layout issues Useful for identifying market requirements Proof of concept	Limited error checking Poor detailed specification to code to Facilitator-driven Limited utility after requirements established Limited usefulness for usability tests Navigational and flow limitations
High-fidelity prototype	Complete functionality Fully interactive User-driven Clearly defines navigational scheme Use for exploration and test Look and feel of final product Serves as a living specification Marketing and sales tool	More resource-intensive to develop Time-consuming to create Inefficient for proof-of-concept designs Not effective for requirements gathering

Case Study 11.1

Paper prototyping as a core tool in the design of cell phone user interfaces

Paper prototyping is being used by cell phone and tablet companies as a core part of their design process (see Figure 11.6). There is much competition in the industry, demanding ever more new concepts. Mobile devices are feature-rich. They include mega-pixel cameras, music players, media galleries, downloaded applications, and more. This requires designing interactions that are complex, but 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 11.6 Prototype developed for cell phone user interface

Mobile device projects involve a range of disciplines – all with their own viewpoint 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 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 US-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 you can experiment with the approach yourself. ■

Although prototypes will have undergone extensive user evaluation, they will not necessarily have been subjected to rigorous quality testing for other characteristics such as robustness 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 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. Throwaway prototyping uses the prototypes as stepping stones towards the final design. In this case, the prototypes are thrown away and the final product is built from scratch. If an evolutionary prototyping approach is to be taken, the prototypes should be subjected to rigorous testing along the way; for throwaway prototyping such testing is not necessary.

DILEMMA

Prototyping versus engineering

Low-fidelity prototypes are not integrated into the final product. In contrast, high-fidelity prototypes can be and so present developers with a dilemma. They can choose to either build the prototype with the intention of throwing it away after it has fulfilled its immediate purpose, or build a prototype with the intention of evolving it into the final product.

The compromises made when producing a prototype must not be ignored – whatever those compromises were. However, when a project team is under pressure, it can become tempting to pull together a set of existing prototypes as the final product. After all, many hours of development will have been spent developing them, and evaluation with the client has gone well, so isn't it a waste to throw it all away? Basing the final product on prototypes in this way will simply store up testing and maintenance problems for later on: in short, this is likely to compromise the quality of the product.

Evolving the prototype into the final product through a defined process of engineering can lead to a robust final product, but this must be clearly planned from the beginning.

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 your market position than having a very high-quality product that reaches the market two months after your competitors'. ■

11.3 Conceptual Design

Conceptual design is concerned with transforming requirements into a conceptual model. Designing the conceptual model is fundamental to interaction design, yet the idea of a conceptual model can be difficult to grasp. One of the reasons for this is that conceptual models take many different forms and it is not possible to provide 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 you with 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 what concepts are needed to understand how to interact with it. The former will emerge from the current functional requirements; possibly it will be a subset of them, possibly all of them, and possibly an extended version of them. The concepts needed to understand how to interact with the product depend on a variety of issues related to 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 getting a concrete view of the conceptual model is to steep yourself in the data you have gathered about your users and their goals and try to empathize with them. From this, a picture of what you want the users' experience to be when using the new product 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. Mood boards (traditionally used in fashion and interior design) may be used to capture the desired feel of a new product (see Figure 11.7). This is informed by results from the requirements activity and considered in the context of technological feasibility.

Figure 11.7 An example mood board

Source: Image courtesy of The Blog Studio www.theblogstudio.com.

There are different ways to achieve empathy with users. For example, Beyer and Holtzblatt (1998), in their method Contextual Design, recommend holding review meetings within the team to get different peoples' perspectives on the data and what they observed. This helps to deepen understanding and to

expose the whole team to different aspects. Ideas will emerge as this extended understanding of the requirements is established, and these can be tested against other data and scenarios, discussed with other design team members, and prototyped for testing with users. Other ways to understand the users' experience are described in Box 11.3.

Key guiding principles of conceptual design are:

- Keep an open mind but never forget the users and their context.
- Discuss ideas with other stakeholders as much as possible.
- Use prototyping to get rapid feedback.
- Iterate, iterate, and iterate.

11.3.1 Developing an Initial Conceptual Model

Some elements of a conceptual model will derive from the requirements for the product. For example, the requirements activity will have provided information about the concepts involved in a task and their relationships, e.g. through task descriptions and analysis. Immersion in the data and attempting to empathize with the users as described above will, together with the requirements, provide information about the product's user experience goals, and give you a good understanding of what the product should be like. In this section we discuss approaches which help in pulling together an initial conceptual model. In particular, we consider:

- Which interface metaphors would be suitable to 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?

BOX 11.3

How to really understand the users' experience

Some design teams go to great lengths to ensure that they come to empathize with the users' experience. This box introduces two examples of this approach.

Buchenau and Suri (2000) describe experience prototyping, which is intended to give designers some insight into a user's experience that can only come from first-hand knowledge. They describe a team designing a

chest-implanted automatic defibrillator. A defibrillator is used with victims of cardiac arrest when their heart muscle goes into a chaotic arrhythmia and fails to pump blood, a state called fibrillation. A defibrillator delivers an electric shock to the heart, often through paddle electrodes applied externally through the chest wall; an implanted defibrillator does this through leads that connect directly to the heart muscle. In either case, it's a big electric shock intended to restore the heart muscle to its regular rhythm that can be powerful enough to knock people off their feet.

This kind of event is completely outside most people's experience, and so it is difficult for designers to gain the insight they need to understand the user's experience. You can't fit a prototype pacemaker to each member of the design team and simulate fibrillation in them! However, you can simulate some critical aspects of the experience, one of which is the random occurrence of a defibrillating shock. To achieve this, each team member was given a pager to take home over the weekend (see Figure 11.8). The pager message simulated the occurrence of a defibrillating shock. Messages were sent at random, 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. Experiences were shared the following week, and 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.

Figure 11.8 The patient kit for experience prototyping

Source: Buchenau, M. and Suri, J. F. (2000) Experience prototyping. In Proceedings of DIS 2000, Design Interactive Systems: Processes, Practices, Methods, Techniques, pp. 17–19.

Another instance is the Third Age suit, an empathy 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 11.9) it has been used to raise awareness within groups of car designers, architects, and other product designers. □

Figure 11.9 The Third Age empathy suit helps designers experience the loss of mobility and sensory perception

Source: Ford Motor Co.

It is not the case that one way of approaching a conceptual design is right for one situation and wrong for another; all of these approaches provide different ways of thinking about the product and help in generating potential conceptual models.

Interface metaphors.

Interface metaphors combine familiar knowledge with new knowledge in a way that will help the user understand the product. Choosing suitable metaphors and combining new and familiar concepts requires a careful balance between utility and fun, and is based on a sound 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 blackboard. But if you consider the users of the system and what is likely to engage them, a metaphor that reminds the children of something they enjoy would be more suitable, such as a ball game, the circus, a playroom, and so on.

Erickson (1990) suggests a three-step process for choosing a good interface metaphor. The first step is to understand what the system will do, i.e. identifying the functional requirements. Developing partial conceptual models and trying them out may be part of the process. The second step is to understand which bits of the product are likely to cause users problems,

i.e. which tasks or subtasks cause problems, are complicated, or are critical. A metaphor is only a partial mapping between the software 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 the tasks is a good starting point. Also, any metaphors used in the application domain with which the users may be familiar may be suitable.

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

1. How much structure does the metaphor provide? A good metaphor will provide structure, and preferably familiar structure.
2. 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.
3. Is the interface metaphor easy to represent? A good metaphor will be associated with particular visual and audio elements, as well as words.
4. Will your audience understand the metaphor?
5. How extensible is the metaphor? Does it have extra aspects that may be useful later on?

For the shared travel organizer introduced in Chapter 10, one metaphor we could use is a printed travel brochure, which is commonly found in travel agents. This familiarity could be combined with facilities suitable for an electronic brochure such as videos of locations, and searching. To evaluate this metaphor, apply the five questions listed above.

1. Does it supply structure? Yes, it supplies structure based on the familiar paper-based brochure. This is a book and therefore has pages, a cover, some kind of binding to hold the pages together, an index, and table of contents. Travel brochures are often structured around destinations but are also sometimes structured around activities, particularly when the company specializes in adventure trips. However, a traditional brochure focuses on the details of the vacation and accommodation and has little structure to support visa or vaccination information (both of which change regularly and are therefore not suitable to include in a printed document).
2. How much of the metaphor is relevant? Having details of the accommodation, facilities available, map of the area, and supporting

illustrations is relevant for the travel organizer, so the content of the brochure is relevant. Also, structuring that information around types of vacation and destinations is relevant, and preferably both sets of grouping could be offered. But the physical nature of the brochure, such as page turning, is less relevant. The travel organizer can be more flexible than the brochure and should not try to emulate its book nature. Finally, the brochure is printed maybe once a year and cannot be kept up-to-date with the latest changes whereas the travel organizer should be capable of offering the most recent information.

3. Is the metaphor easy to represent? Yes. The vacation information could be a set of brochure-like pages. Note that this is not the same as saying that the navigation through the pages will be limited to page-turning.
4. Will your audience understand the metaphor? Yes.
5. How extensible is the metaphor? The functionality of a paper-based brochure is fairly limited. However, it is also a book, and we could borrow facilities from ebooks (which are also familiar objects to most of our audience), so yes, it can be extended.

Activity 11.2

Another possible interface metaphor for the travel organizer is the travel consultant. A travel consultant takes a set of requirements and tailors the vacation accordingly, offering maybe two or three alternatives, but making most of the decisions on the travelers' behalf. Ask the five questions above of this metaphor.

Comment

Show/Hide

Interaction types.

Chapter 2 introduced four different types of interaction: instructing, conversing, manipulating, and exploring. Which 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 drawing package has aspects of instructing and conversing.

Most conceptual models will include a combination of interaction types, and it

is necessary to associate different parts of the interaction with different types. For example, in the 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, e.g. 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. (You may like to refer back to the scenario of this task in Chapter 10 and consider how well it matches this type of interaction.) Alternatively, for users who don't have any clear requirements yet, they might prefer to explore availability before asking for specific options.

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 the product under development and suggest different possible behaviors. Therefore considering the effect of different interfaces on the product at this stage is one way to prompt alternatives.

Before the product can be prototyped, some candidate alternative interfaces will need to have been chosen. These decisions will depend on the product constraints, arising from the requirements you have established. For example, input and output devices will be influenced particularly by user and environmental requirements. Therefore, considering interfaces here also takes one step towards producing practical prototypes.

To illustrate this, we consider a subset of the interfaces introduced in Chapter 6, and the different perspectives they bring to the 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 6 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.
- Tangible interface. Tangible interfaces are a form 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 traveling, but there are practical problems of having this kind of interface, as the objects may be lost or damaged.

- Augmented and mixed reality. The travel organizer is not the kind of product that is usually designed for an augmented or mixed reality interface. The question is what would the physical object be in this case, that the virtual element could enhance? One possibility might be to enhance the physical brochure to provide more dynamic and easily changed information.

Activity 11.3

Consider the movie rental subscription service introduced in Chapter 10.

1. Identify tasks associated with this product that would best be supported by each of the interaction types instructing, conversing, manipulating, and exploring.
2. Pick out two interface types from Chapter 6 that might provide a different perspective on the design.

Comment

Show/Hide

11.3.2 Expanding the Initial Conceptual Model

Considering the issues in the previous section helps the designer to produce a set of initial conceptual model ideas. These ideas must be thought through in more detail and expanded before being prototyped or tested with users. For example, concrete suggestions of the concepts to be communicated between the user and the product and how they are to be structured, related, and presented are needed. This means deciding 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. These decisions will be made initially only tentatively and may change after prototyping and evaluation.

What functions will the product perform?

Understanding the tasks the product will support is a fundamental aspect of developing the conceptual model, but it is also important to consider which elements of the task will be the responsibility of the user and which will be carried out by the product. For example, the travel organizer may suggest specific vacation options for a given set of people, but is that as much as it should do? Should it automatically reserve the booking, or wait until it is told that this travel arrangement is suitable? Developing scenarios, essential use cases, and use cases will help clarify the answers to these questions. Deciding what the system will do and the user will do is sometimes called task allocation. This trade-off has cognitive implications (see Chapter 3), and is linked to social aspects of collaboration (see Chapter 4). 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, and thereby indirectly in the control of the human user.

How are the functions related to each other?

Functions may be related temporally, e.g. one must be performed before another, or two can be performed in parallel. They may also be related through any number of possible categorizations, e.g. all functions relating to privacy on a smartphone, or all options for viewing photographs in 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 task analysis has been performed, the breakdown will support these kinds of decision. For example, in the travel organizer example, the task analysis performed in Section 10.7 shows the subtasks involved and the order in which the subtasks can be performed. Thus, the system could allow potential travel companies to be found before or after investigating the destination's facilities. It is, however, important to identify the potential travel companies before looking for availability.

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 necessary 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 set of people using the travel organizer requires the following information: 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 any special requirements (such as disability) that this group has. In order 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 may show a part of the product or a complete overview. Case Study 11.2 and Chapter 12 include more information and some examples.

11.4 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 wishes 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 6 introduces several interface types and their associated design issues; these issues represent the kinds of decision that need to be made during concrete design. Case study 11.2 illustrates the impact that different-sized devices may have on the same application, and the need to explicitly design for

different form factors. Chapter 6 also introduces some guidelines, principles, and rules for different interface types to help designers ensure that their products meet usability and user experience goals.

Case Study 11.2

Designing mobile applications for multiple form factors

Trutap is a social networking service designed for more than 350 different models of mobile device, which was built for a UK startup between 2007 and 2009. It aggregates online blogging, instant messaging, and social services like Facebook, allowing its users to interact with them (see Figures 11.10 and 11.11). The design of the Trutap application, which took place over two major releases, posed significant challenges in terms of how to integrate disparate sources of data onto small-screen devices, and produce a design which would scale between form factors, i.e. different physical handset designs.

Figure 11.10 Trutap: version 2.0 design concepts

Source: © Trutap, Reproduced with permission.

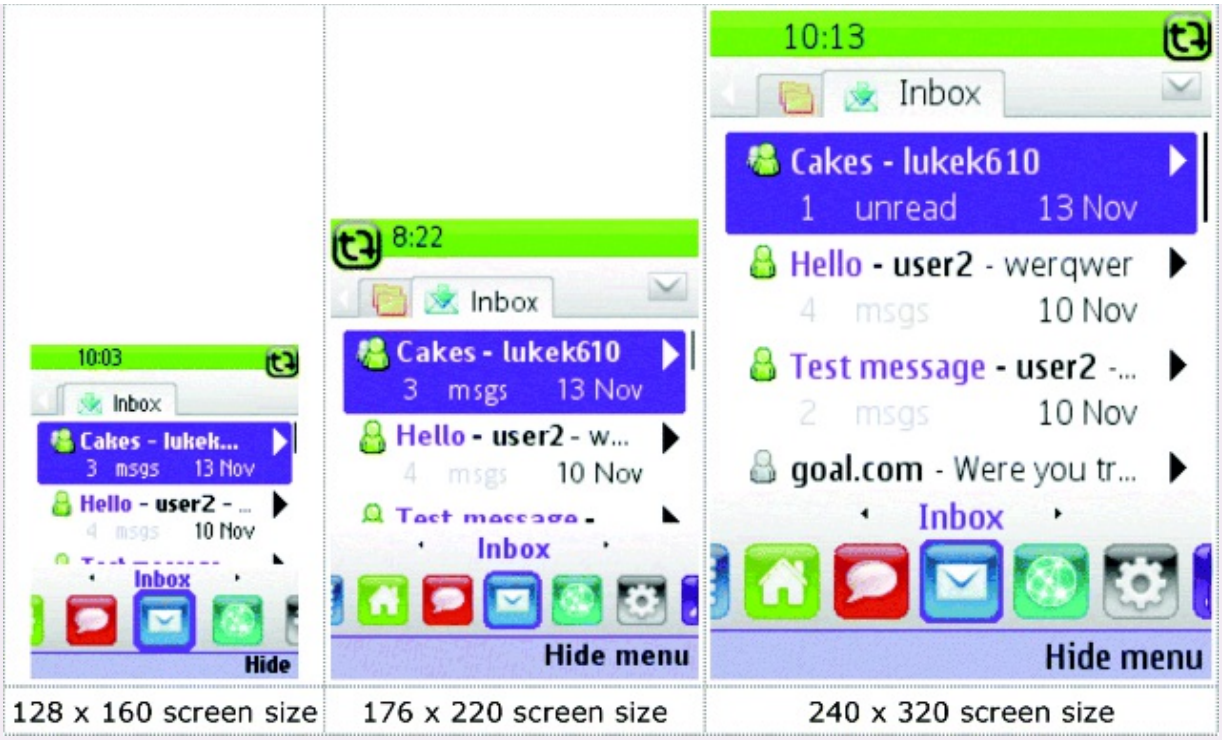

Figure 11.11 Trutap: version 2.0 screenshots, inbox

Source: © Trutap, Reproduced with permission.

The product was designed with a clear goal: teenagers and young adults were spending much of their social lives online. Trutap would help them keep connected, wherever they were.

Two versions of the product were launched: Trutap 1.0 offered its own mechanisms for managing people's contacts and communicating with them, and tied into a range of existing instant messaging networks (Yahoo!, MSN, AOL, and the like). Launched in 2008, this version saw far greater take-up in India and Indonesia than with its original target audience of UK students.

This take-up, combined with the successful launch of the iPhone in July 2008 and the increasing prominence of Facebook as the dominant site for personal social networking, led to a change in emphasis for the 2.0 release of Trutap. Launched a year after 1.0, and technically an evolution rather than a reworking, release 2.0 emphasized the aggregation of existing online services, tying into Facebook, weblogging software, and photo management, and extending the number of instant messaging services covered. Publicly, the product was presented as a means for aspirational middle classes in the developing world to experience many of the same benefits that the iPhone promised, but on their conventional mobile devices.

This case study, by Tom Hume, Johanna Hunt, Bryan Rieger, and Devi

Lozdan from Future Platforms Ltd, explores the impact that different form factors had on the design of Trutap. ■

Concrete design also deals with issues related to user characteristics and context, and two aspects that have drawn particular attention for concrete design are accessibility and national culture. Accessibility was discussed in Box 1.2. Researchers, designers, and evaluators have investigated a range of techniques, toolkits, and interaction devices to support individuals with different accessibility needs. More recently, there has been a reaction to this approach that challenges the ‘rhetoric of compassion’ in favor of a ‘rhetoric of engagement', and suggests that users be empowered rather than designed for (Rogers and Marsden, 2013).

Aspects of cross-cultural design include use of appropriate language(s), colors, icons and images, navigation, and information architecture (Rau et al, 2013). Example design guidelines include ensuring that the product supports different formats for dates, times, numbers, measurements, and currencies, and that generic icons are designed where possible (Esselink, 2000). However, 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 (see also Case Study 11.3 and Gary Marsden's interview at the end of this chapter).

Guidelines, although seemingly attractive, can be misguided. One of the most well-known sets of guidelines for cultural web design was proposed by Marcus and Gould (2000), building on the cultural dimensions proposed by Hofstede (1994). However, Hofstede's work, and its application in interaction design, has been challenged (see Box 11.4), and designing for a cross-cultural audience is now recognized as more than a translation exercise. As Carlson (1992, p. 175) has put it, successful products “are not just bundles of technical solutions; they are also bundles of social solutions. Inventors succeed in a particular culture because they understand the values, institutional arrangements, and economic notions of that culture.”

BOX 11.4

Using Hofstede's dimensions in interaction design

One of the most influential pieces of work on characterizing national culture differences was carried out by a management theorist called Geert Hofstede around 1970. He was given access to responses from a survey of IBM employees in over 50 countries worldwide and from this he identified four dimensions of national culture: power distance (PD), individualism (IND), masculinity–femininity (MAS), and uncertainty avoidance (UA). As a result of work done in Hong Kong at a later date by a Canadian, Michael Bond, a fifth dimension was added that deals with time orientation.

Although influential, Hofstede's work does have limitations. For example, he admits that the people involved in designing the original questionnaire were all from Western cultures. In addition, his studies have been discussed and challenged over the intervening years: e.g. Oyserman et al (2002) challenged his finding that European Americans are more individualistic than people from other ethnic groups. The application of his ideas in interaction design has also been challenged – e.g. work by Oshlyansky (2007) found that Hofstede's model does not help explain cultural differences in affordance; nor does it seem to apply to technology acceptance. So, although popular, Hofstede's dimensions may not be the best approach to accommodating national culture differences in interaction design. □

11.5 Using Scenarios

Scenarios are informal stories about user tasks and activities. Scenarios can be used to model existing work situations, but they are more commonly used for expressing proposed or imagined situations to help in conceptual design. Often, stakeholders are actively involved in producing and checking through scenarios for a product. Bødker suggests four roles (Bødker, 2000, p. 63):

1. As a basis for the overall design.
2. For technical implementation.
3. As a means of cooperation within design teams.
4. As a means of cooperation across professional boundaries, i.e. as a basis of communication in a multidisciplinary team.

In any one project, scenarios may be used for any or all of these. More specifically, scenarios have been used as scripts for user evaluation of prototypes, as the basis of storyboard creation (see Section 11.6.1), and to build a shared understanding among team members. Scenarios are good at selling ideas to users, managers, and potential customers.

Bødker 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 (see Figure 11.12), thereby helping designers to gain a more comprehensive view of the proposal. This idea has been extended by Mancini et al (2010) who use positive and negative video scenarios to explore futuristic technology.

Scenario 3: Hyper-wonderland

This scenario addresses the positive aspects of how a hypermedia solution will work.

The setting is the Lindholm construction site sometime in the future.

Kurt has access to a portable PC. The portables are hooked up to the computer at the site office via a wireless modem connection, through which the supervisors run the hypermedia application.

Action: During inspection of one of the caissons[1] Kurt takes his portable PC, switches it on and places the cursor on the required information. He clicks the mouse button and gets the master file index together with an overview of links. He chooses the links of relevance for the caisson he is inspecting.

Kurt is pleased that he no longer needs to plan his inspections in advance. This is a great help because due to the 'event-driven' nature of inspection, constructors never know where and when an inspection is taking place. Moreover, it has become much easier to keep track of personal notes, reports etc. because they can be entered directly on the spot.

The access via the construction site interface does not force him to deal with complicated keywords either. Instead, he can access the relevant information right away, literally from where he is standing.

A positive side-effect concerns his reachability. As long as he has logged in on the computer, he is within reach of the secretaries and can be contacted when guests arrive or when he is needed somewhere else on the site. Moreover, he can see at a glance where his colleagues are working and get in touch with them when he needs their help or advice.

All in all, Kurt feels that the new computer application has put him more in control of things.

[1] Used in building to hold water back during construction.

Scenario 4: Panopticon

This scenario addresses the negative aspects of how a hypermedia solution will work.

The setting is the Lindholm construction site sometime in the future.

Kurt has access to a portable PC. The portables are hooked up to the computer at the site office via a wireless modem connection, through which the supervisors run the hypermedia application.

Action: During inspecting one of the caissons Kurt starts talking to one of the builders about some reinforcement problem. They argue about the recent lab tests, and he takes out his portable PC in order to provide some data which justify his arguments. It takes quite a while before he finds a spot where he can place the PC: either there is too much light, or there is no level surface at a suitable height. Finally, he puts the laptop on a big box and switches it on. He positions the cursor on the caisson he is currently inspecting and clicks the mouse to get into the master file. The table of contents pops up and from the overview of links he chooses those of relevance – but no lab test appears on the screen. Obviously, the file has not been updated as planned.

Kurt is rather upset. This loss of prestige in front of a contractor engineer would not have happened if he had planned his inspection as he had in the old days.

Sometimes. he feels like a hunted fox especially in situations where he is drifting around thinking about what kind of action to take in a particular case. If he has forgotten to log out, he suddenly has a secretary on the phone: “I see you are right at caisson 39, so could you not just drop by and take a message?”

All in all Kurt feels that the new computer application has put him under control.

Figure 11.12 Example plus and minus scenarios

Source: S. Bødker (2000) Scenarios in user-centered design – setting the stage for reflection and action. Interacting with Computers, 13(1), Fig. 2, p. 70.

11.6 Generating Prototypes

In this section we illustrate how prototypes may be used in design, and demonstrate one way in which prototypes may be generated from the output

of the requirements activity: producing a storyboard from a scenario and a card-based prototype from a use case. Both of these are low-fidelity prototypes and they may be used as the basis to develop more detailed interface designs and higher-fidelity prototypes as development progresses.

11.6.1 Generating Storyboards from Scenarios

A storyboard represents a sequence of actions or events that the user and the product go through to achieve a task. A scenario is one story about how a product may be used to achieve the task. It is therefore possible to generate a storyboard from a scenario by breaking the scenario into a series of steps which focus on interaction, and creating one scene in the storyboard for each step. The purpose for doing this is two-fold: first, to produce a storyboard that can be used to get feedback from users and colleagues; 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 10. This can be broken down into five main steps:

1. The Thomson family gather around the organizer and enter a set of initial requirements.
2. The system's initial suggestion is that they consider a flotilla trip but Sky and Eamonn aren't happy.
3. The travel organizer shows them some descriptions of the flotillas written by young people.
4. Will confirms this recommendation and asks for details.
5. The travel organizer emails the details.

Activity 11.4

Consider an augmented reality in-car navigation system that takes information from a GPS and displays routes and traffic information directly onto the car windscreen. Suggest one plus and one minus scenario. For the plus scenario, think of the possible benefits of the system. For the minus scenario, imagine what could go wrong.

Figure 11.13 An example car-navigation system based on augmented reality

Source: The Aeon Project, courtesy of Michaël Harboun

Comment

Show/Hide

The first thing to notice about this set of steps is that it does not have the detail of a use case but identifies the key events or activities associated with the scenario. The second thing to notice is that some of these events are focused solely on the travel organizer's screen and some are concerned with the environment. For example, the first one talks about the family gathering around the organizer, while the third and fifth are focused on the travel organizer. We therefore could produce a storyboard that focuses on the screens or one that is focused on the environment. Either way, sketching out the storyboard will prompt us to think about design issues.

For example, the scenario says nothing about the kind of input and output devices that the system might use, but drawing the organizer forces you to think about these things. There is some information about the environment within which the system will operate, but again drawing the scene makes you

stop and think about where the organizer will be. You don't have to make any decisions about, e.g. using a trackball, or a touch screen, or whatever, but you are forced to think about it. 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 is also made more explicit because of the drawing act.

We chose to draw a storyboard that focuses on the environment of the travel organizer, and it is shown in Figure 11.14. While drawing this, various questions relating to the environment came to mind such as how can the interaction be designed for all the family? Will they sit or stand? How confidential should the interaction be? What kind of documentation or help needs to be available? What physical components does the travel organizer need? And so on. In this exercise, the questions it prompts are just as important as the end product.

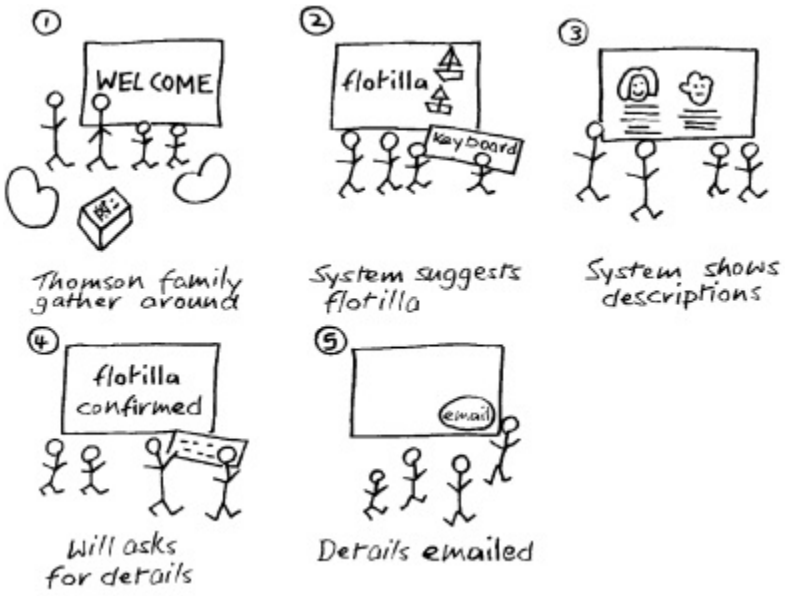

Figure 11.14 The storyboard for the travel organizer focusing on environmental issues

Note that although we have used the scenario as the main driver for producing the storyboard, there is other information from the requirements activity that also informs development.

Activity 11.5

Activity 10.3 asked you to develop 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 are drawing this storyboard, write down the design issues you are prompted to consider.

Comment

Show/Hide

11.6.2 Generating Card-Based Prototypes from Use Cases

The value of a card-based prototype lies in the fact that the screens or 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. Where a storyboard focusing on the screens has been developed, this can be translated 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 case generated for the travel organizer in Section 10.6.2. This focused on the visa requirements part of the system. For each step in the use case, the travel organizer will need to have an interaction component to deal with it, e.g. a button or menu option, or a display. By stepping through the use case, it is possible to build up a card-based prototype to cover the required behavior. For example, the cards in Figure 11.16 were developed by considering each of the steps in the use case. Card one covers step 1; card two covers steps 2, 3, 4, 5, 6, and 7; and card three covers steps 8, 9, 10, and 11 (notice the print button that is drawn into card three to allow for steps 10 and 11). As with the storyboards, drawing concrete elements of the interface like this forces the designer to think about detailed issues so that the user can interact with the prototype. In card two you will see that I chose to use a drop-down menu for the country and nationality. This is to avoid mistakes. However, the flaw in this is that I may not catch all of the countries in my list, and so an alternative design could also be incorporated where the user can choose an ‘enter below’ option and then type in the country or nationality (see Figure 11.17).

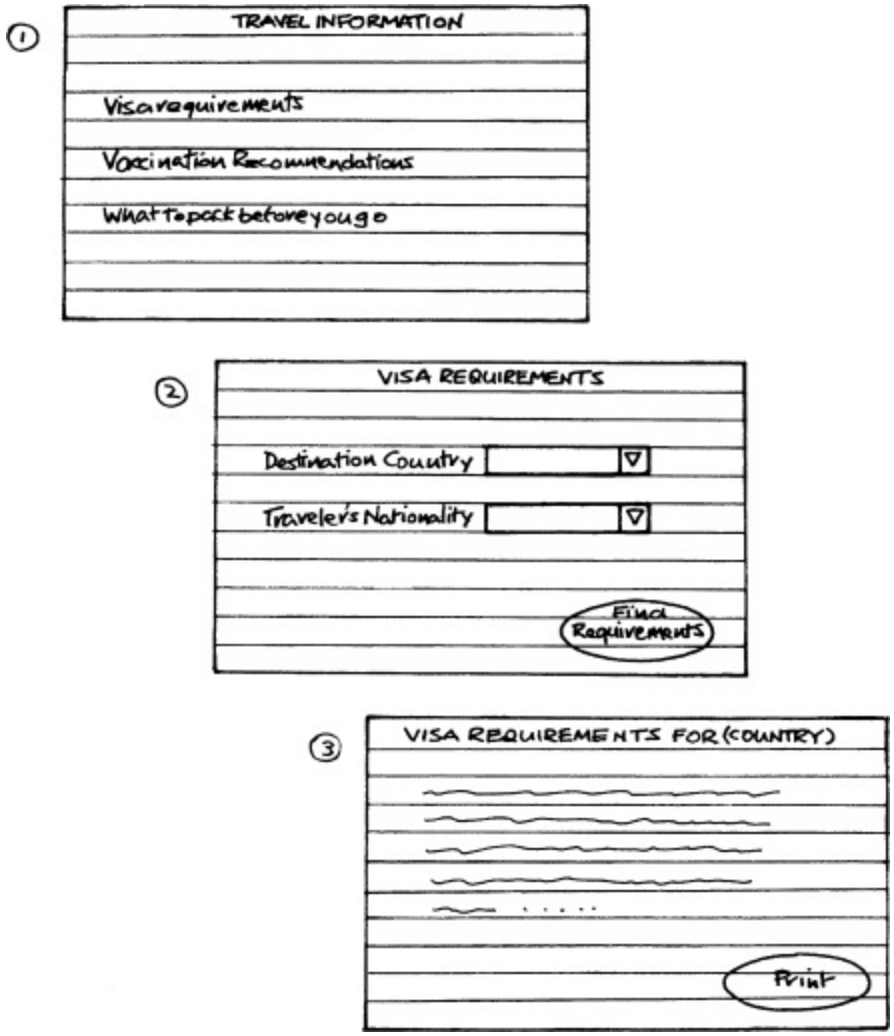

Figure 11.16 Cards one to three of a card-based prototype for the travel organizer

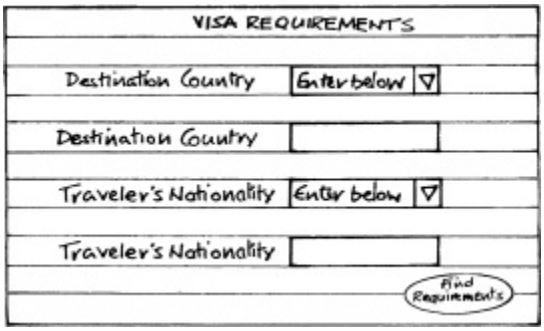

Figure 11.17 Card four of a card-based prototype for the travel organizer

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

Activity 11.6

Produce a card-based prototype for the movie rental subscription service and the task of renting a movie as described by the use case in Activity 10.4. You may also like to ask one of your peers to act as a user and step through the task using the prototype.

Comment

Show/Hide

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 achieved by creating a visual representation of the user's experience. These representations are variably called a design map (Adlin and Pruitt, 2010) or a customer journey map (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, 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, e.g. for a flight (see Figure 11.19(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 in Figure 10.4(b)). Figure 11.19(b) illustrates the structure of a timeline and how different kinds of issues may be captured, e.g. questions, comments, and ideas.

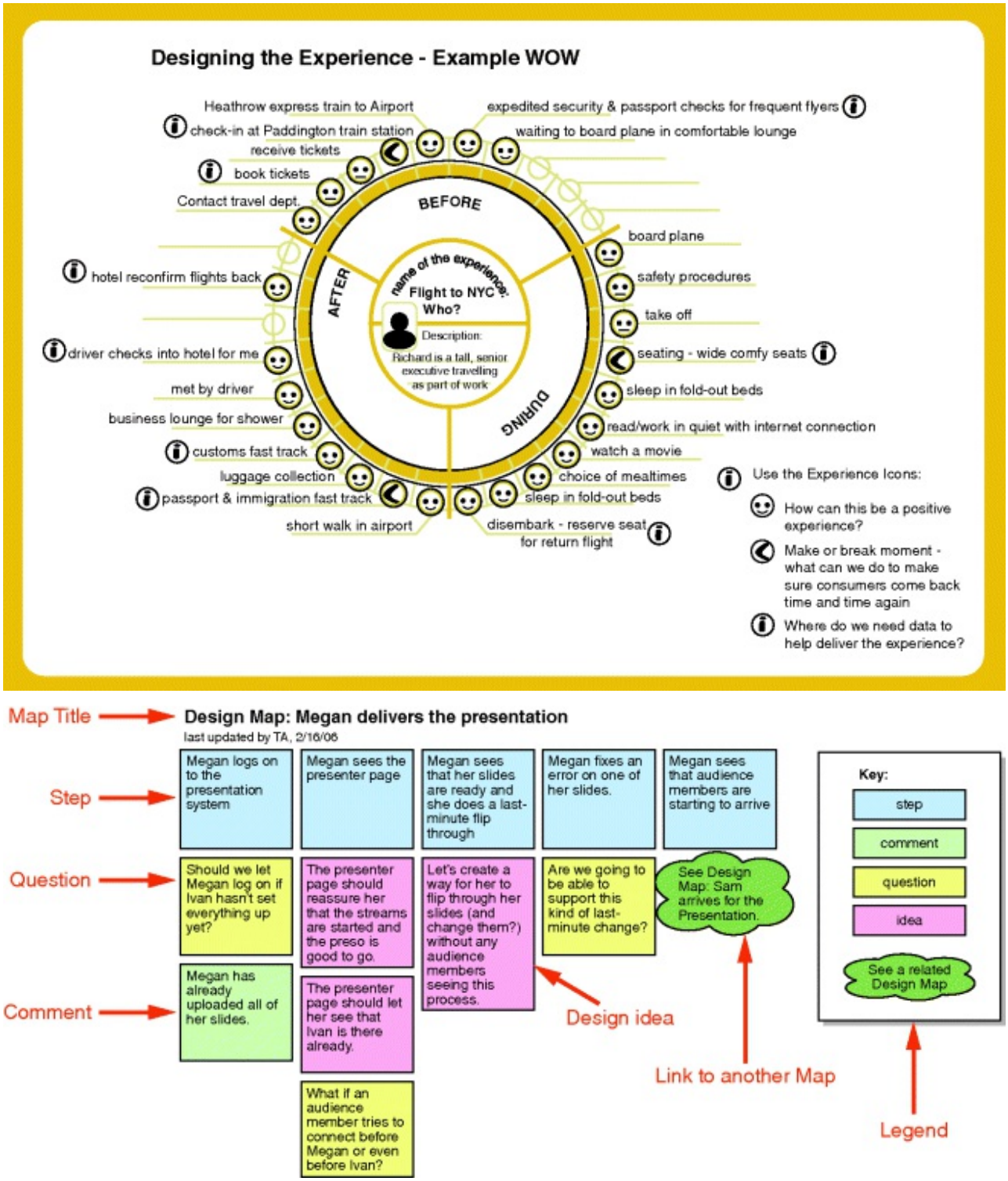

Figure 11.19 (a) An experience map using a wheel representation. (b) An example timeline design map illustrating how to capture different issues.

Source: (a) http://www.ux-lady.com/experience-maps-user-journey-and-more-exp-map-layout/ (b) Adlin, T. and Pruitt, J. (2010) The Essential Persona Lifecycle: Your guide to building and using personas. Morgan Kaufmann p. 134.

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 sometimes this can be used to identify missing functionality or areas of under-designed interaction.

Video illustrating the benefits of experience mapping using a timeline at http://youtu.be/eLT_Q8sRpyI

BOX 11.5

Involving users in design: participatory design

The idea of participatory design 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. A fuller history of the movement is given in Ehn (1989) and Nygaard (1990).

Several projects at this time attempted to involve users in design and focus on work rather than on simply producing a product. One of the most discussed is the UTOPIA project, a cooperative effort between the Nordic Graphics Workers Union and research institutions in Denmark and Sweden to design computer-based tools for text and image processing.

Involving users in design decisions is not simple, however. Cultural differences can become acute when users and designers are asked to work together to produce a specification for a system. Bødker et al (1991) recount the following scene from the UTOPIA project: “Late one afternoon, when the designers were almost through with a long presentation of a proposal for the user interface of an integrated text and image processing system, one of the typographers commented on the lack of information about typographical code-structure. He didn't think that it was a big error (he was a polite person), but he just wanted to point out that the computer scientists who had prepared the proposal had forgotten to specify how the codes were to be presented on the screen. Would it read ‘<bf/’ or perhaps just ‘\b’ when the text that followed was to be printed in boldface?”

In fact, the system being described by the designers was a WYSIWYG

(what you see is what you get) system, and so text that needed to be in bold typeface would appear as bold (although most typographic systems at that time did require such codes). The typographer was unable to link his knowledge and experience with what he was being told. In response to this kind of problem, the project started using mockups. Simulating the working situation helped workers to draw on their experience and tacit knowledge, and designers to get a better understanding of the actual work typographers needed to do.

Case Study 11.3 describes an extension to the participatory design idea, called community-based design. □

Case Study 11.3

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 to both 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 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.

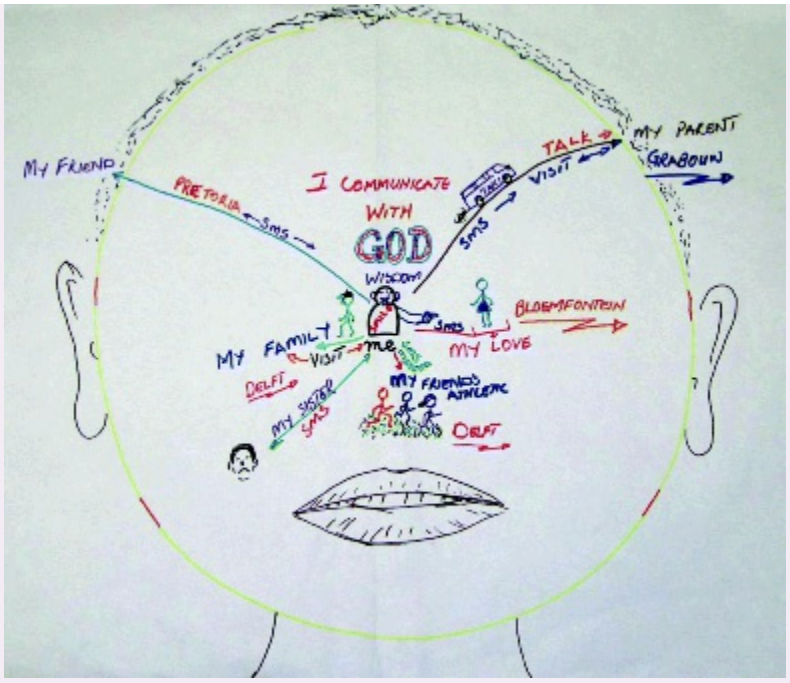

Figure 11.20 One participant's view of communication

Source: Copyright Edwin Blake et al.

Figure 11.21 Participants discussing design in sign language

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 than 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, 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. ■

11.7 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, or a piece of software, or both. Whatever the final form, it is very unlikely that you will develop anything from scratch as there are many useful (in some cases essential) resources to support development. Here we introduce two kinds of resource: physical computing kits and software development kits (SDKs).

11.7.1 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 (e.g. accelerometers, infrared, temperature) to detect states, and actuators (e.g. motors, valves) 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 catflap. The movement triggers an actuator to take a photo of what came through the catflap using a webcam positioned on the back door. The photo is uploaded to a website that alerts the owner if there are cats or other objects that do not match their own cat's image.

A number of physical computing toolkits have been developed for educational and prototyping purposes. One of the earliest is Arduino (see Banzi, 2009). The aim 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 11.22), which is the piece of hardware that is used to build objects, and the Arduino IDE (integrated development environment), 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 a number of protruding 'legs' that provide input and output pins – which the sensors and actuators are connected to. Sketches are written in the IDE using a simple processing language, then uploaded to the board and translated into the 'C' programming language.

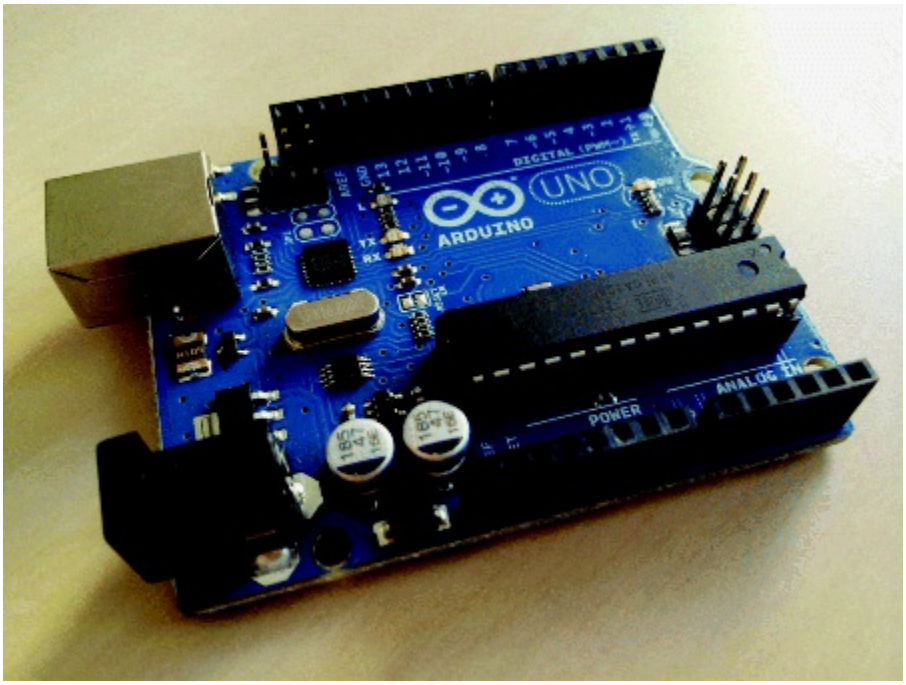

Figure 11.22 The Arduino board

Source: Courtesy of 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 11.23 and her interview at the end of Chapter 6). 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 multicolour 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 11.23 The Lilypad Arduino kit

Source: Photo courtesy of Leah Buechley http://web.media.mit.edu/~leah/LilyPad/build/turn_signal_jacket.html.

Video introducing MakeMe (Marquardt et al, 2015), a novel toolkit that is assembled from a flat electronic sheet, where six sides are snapped out then 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 **http://www.codeme.io/**

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 with – are Senseboard (Richards and Woodthorpe, 2009), LittleBits, and MaKey MaKey (Silver and Rosenbaum, 2012). The MaKey MaKey toolkit comprises a printed circuit board with an Arduino microcontroller, alligator clips, and a USB cable (see Figure 11.24). 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 that alligator clips are clipped onto in order to connect with a computer via the USB cable. The other ends of the clips can be attached to any non-insulating 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.

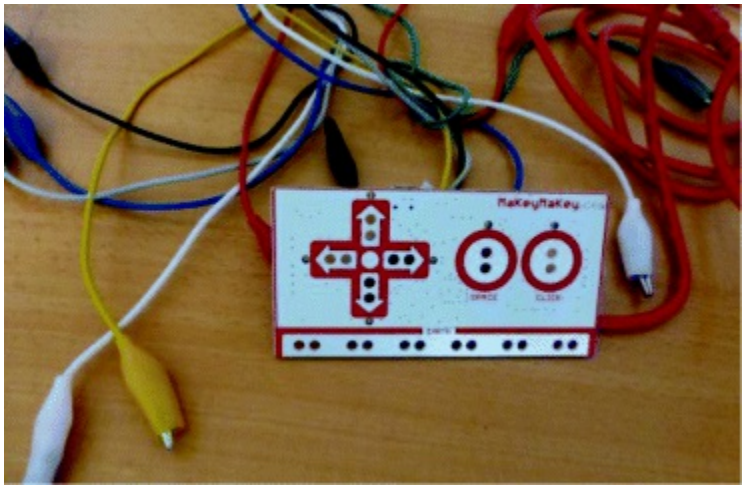

Figure 11.24 The MaKey MaKey toolkit

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 (e.g. Hodges et al, 2013). However, 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 early 60s and late 80s, assembled and played with the MaKey MaKey toolkit (see Figure 11.25). 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 around.

Figure 11.25 A group of retired friends playing with a MaKey MaKey toolkit

BOX 11.6

The rise of the Maker Movement

The Maker Movement emerged in the mid 2000s. Following in the footsteps of the computer revolution and the Internet, it is viewed as the next big revolution that will transform 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 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 makespaces. In a nutshell, it is about inventing the future through connecting technologies, the Internet, and physical things.

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 ground swell in interest in how to make. 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 – which would have been impossible for them to do

previously – such as large 3D models, furniture, and installations. Then smaller makerspaces started appearing in their 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 1.3). 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 11.2.4) 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 a popular online marketplace for people who make things to sell their crafts and other handmade items. It is designed to be easy for makers to use and set up their store to sell to family, friends, and strangers across the world. Unlike the corporate online sites, (e.g. Amazon, eBay), it is a place for craft makers to reach others and to show off their wares in ways they feel best fit what they have made.

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). □

Activity 11.7

Watch the video of Lady Gaga in the Voltanis, the first flying dress, developed by the e-textile company XO. What do you think of this fusion of fashion and state-of-the-art electronics and technology?

Video of Lady Gaga in the Voltanis at **http://vimeo.com/91916514**

Comment

Show/Hide

11.7.2 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, e.g. for iOS on iPad, iPhone, and iPod touch, for the Kinect device, and for the Windows phone. Typically an SDK includes an IDE (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 so much easier using such a powerful resource, and so much more can be achieved.

For example, the availability of Microsoft's Kinect SDK makes the device's powerful gesture recognition and body motion tracking capabilities accessible. This has led to the exploration of, for example, motion tracking in immersive games (Manuel et al, 2012), user identification using body lengths (Hayashi et al, 2014), and robot control (Wang et al, 2013).

An SDK will include a set of application programming interfaces (APIs) that allows control of the components without knowing the intricacies of how they work. In some cases, access to the API alone is sufficient to allow significant work to be udertaken, e.g. Hayashi et al (2014) only needed access to the APIs. The difference between APIs and SDKs is explored in Box 11.7.

BOX 11.7

APIs and SDKs

SDKs (software development kits) consist of a set of programming tools and components while an API (application programming interface) is the set of inputs and outputs, i.e. 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 fancy, rather than using pre-shaped 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 the relevant APIs, but it adds programming tools, documentation, and other development support as well. □

Assignment

This assignment continues work on the online booking facility introduced at the end of Chapter 10. The work will be continued in the assignments for Chapters 12, 14, and 15.

a. Based on the information gleaned from the assignment in Chapter 10, suggest three different conceptual models for this system. You should 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.

b. Produce the following prototypes for your chosen conceptual model:

   i. Using the scenarios generated for the online booking facility, produce a storyboard for the task of booking a ticket for one of your conceptual models. Show it to two or three potential users and get some informal feedback.

   ii. Now develop a card-based prototype from the use case for the task of booking a ticket, also incorporating feedback from part (i). Show this new prototype to a different set of potential users and get some more informal feedback.

c. Consider your product's concrete design. Sketch out the application's landing page. Consider the layout, use of colors, navigation, audio, animation, etc. While doing this, use the three main questions introduced in Chapter 6 as guidance: Where am I? What's here? Where can I go? Write one or two sentences explaining your choices, and consider whether the choice is a usability consideration or a user experience consideration.

d. Sketch out an experience map for your product. Use the scenarios and personas you have already generated to explore the user's experience. In particular, identify any new interaction issues that you had not considered before, and suggest what you could do to address them.

e. How does your 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 do you think not?

Take a Quickvote on Chapter 11:
www.id-book.com/quickvotes/chapter11

Summary

This chapter has explored the activities of design, prototyping, and construction. Prototyping and scenarios are used throughout the design process to test out 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 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 edn). 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.

CARROLL, J. M. (ed.) (1995) Scenario-based Design. John Wiley & Sons, Inc. This volume is an edited collection of papers arising from a three-day workshop on use-oriented design. The book contains a variety of papers including case studies of scenario use within design, and techniques for using them with object-oriented development, task models, and usability engineering. This is a good place to get a broad understanding of this form of development.

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

LAZAR, J. (ed.) (2007) Universal Usability: Designing information systems for diverse user populations. John Wiley & Sons Ltd. This book provides an interesting selection of case studies that demonstrate how developers can design for diverse populations to ensure universal usability.

Interview with the late Gary Marsden

Gary Marsden died suddenly and unexpectedly in December 2013. He was only 43. He was a professor in the Computer Science Department at the University of Cape Town. His research interests spanned mobile interaction, computer science, design and ICT for Development. He is a co-author of a book published in 2015, with Matt Jones and Simon Robinson, entitled, There's Not an App for That: Mobile User Experience Design for Life. He was also a co-author of Mobile Interaction Design, which was published in 2006. He won the 2007 ACM SIGCHI Social Impact Award for his research in using mobile technology in the developing world. He made a big impression on the HCI world. We have decided to keep his interview from the 3rd edition.

Gary, can you tell us about your research and why you do it?

My work involves creating digital technology for people living in Africa. Most of this work is based on designing software and interfaces for mobile cellular handsets as this is currently the most prevalent digital technology within Africa.

Because the technology is deployed in Africa, we work within a different design space than those working in more developed parts of the world. For instance, we assume that users have no access to personal computers or high-speed Internet connections. We must also take into account different literacy levels in our users and the cultures from which they come. Not only does this affect the technology we create, but the

methods we use to create it.

As a computer science professional, I want to understand how to create digital systems that are relevant and usable by the people purchasing them. For many people here, buying a cellular handset is a significant investment and I want to make sure that the discipline of interaction design is able to help deliver a product which maximizes the purchaser's investment.

How do you know if the systems that you build are what people want and need?

This is currently a hotly debated topic in the field and it is only recently that there has been sufficient work from which to draw conclusions.

The first challenge crops up in designing a system for people who have very little exposure to technology. For many of our users, they have no experience of digital technology beyond using a simple cellular handset. Therefore, participatory techniques, where users are asked to become co-designers, can be problematic as they have no abstract notions of basic ideas like the separation between hardware and software. To overcome this, we often take a technology probe approach, allowing users to comment on a high-fidelity prototype rather than require them to make abstract decisions about a series of paper sketches.

For many of the systems we build, we are interested in more than simple measures of efficiency and effectiveness. Sure, it is important that technology is usable, but in the resource-constrained environment, it is critical that the technology is useful; money is too scarce to spend on something that does not significantly improve livelihood.

To measure impact on people and communities we often borrow from the literature on development and measure issues like domestification – the extent to which a technology is appropriated into someone's day-to-day living. In a lot of our work we also partner with non-governmental organizations (NGOs) who are based in a community and are looking for research partners to provide digital solutions to problems they meet – for instance, we have worked with a voter education NGO that wanted to use digital technology to better inform voters about their choices in an upcoming election. In that project we would adopt the goals of the NGO (how much people understand their voting choices) as part of the success criteria for our project. Often NGOs have sophisticated instruments to measure the impact they are having, as their funding relies on it. We can use those instruments to measure our impact.

To understand how our participants truly feel about a system, we use ‘polyphonic’ assessment, as reported by Bill Gaver. The method employs unbiased journalists who interview users and report their assessment of the system. We have adopted this approach in our work and found it to be highly effective in gaining feedback on our systems. Furthermore, it overcomes a strong Hawthorne effect experienced by researchers who work in resource poor environments – users are so grateful for the attention and resources being given them, they rate any system highly in an attempt to please the researchers and keep them investing in that community.

At present, there is no clear consensus about how best to evaluate technology deployments in developing world communities, but it is clear that the technology cannot be evaluated solely on a human–computer interaction level, but needs to be considered on a livelihoods and community impact level.

Have you encountered any big surprises in your work?

My work seems to be endlessly surprising which, as a researcher, is highly stimulating. The first surprise when I moved here 12 years ago, was the penetration of mobile handsets. In an era when handsets were considered a luxury in Europe (1999), I saw people living in shacks talking on their mobile handsets. Clearly domestification was not an issue for cellular technology.

When I started to run research projects in Africa, I was surprised by the extent to which much HCI research and methods incorporated assumptions based in the developed world – for example, the issue I mentioned earlier around participatory design. Also, the early HCI literature I read on the internationalization of interfaces did not stand me in good stead. For example, my colleague, Marion Walton, built one interface consisting of a single button on a screen. We asked participants to click on the button, but one participant was unable to do this. When we pointed out the button to him, he said, ‘That is not a button, that is a picture of a button.’ Of course, he was correct and we learnt something valuable that day about visual culture.

Finally, the environment in Africa leads to surprises. The strangest problem I have had was trying to fix a computer in rural Zambia that had suddenly stopped working. On taking the casing off, I discovered white ants had eaten the green resin out of the circuit board and used it to build a nest over the power supply (where it was warm). Although it now looked like a beautiful lace, the motherboard could not be salvaged.

What are your hopes for the future?

My hope and my passion are to create a new generation of African computer scientists who create technology for their continent. Whilst the work I am engaged in may be helping to some small degree, it is not sustainable for outside people or teams to create new technology for everyone who lives in the developing world. As an educator, I believe the solution is to teach interaction design in African universities and empower Africans to create the technology that is most appropriate to them and their environment. □