CHAPTER

Design Production

9

Objectives

After reading this chapter, you will:

1. Know how to use requirements to drive design
2. Understand the macro view of lifecycle iteration for design
3. Be able to unpack conceptual designs and explore strategies for realization in intermediate design
4. Understand wireframes and how to make and use them
5. Be prepared to use annotated scenarios, prototypes, and wireframes to represent screens and navigation in detailed design
6. Know how to maintain a custom style guide in design
7. Understand the concept of interaction design specifications for software implementation

9.1 INTRODUCTION

9.1.1 You Are Here

We begin each process chapter with a "you are here" picture of the chapter topic in the context of the overall Wheel lifecycle template; see Figure 9-1. This chapter is a continuation of the previous one about designing the new work practice and the new system.

In Chapter 7 we did ideation and sketching and in Chapter 8 we conceptualized design alternatives. Now it is time to make sure that we account for all the requirements and envisioned models in those designs. This is especially important for domain-complex systems where it is necessary to maintain connections to contextual data.

The translation from requirements to design is often regarded as the most difficult step in the UX lifecycle process. We should expect it to be difficult because now that we have made the cognitive shift from analysis-mode thinking to synthesis-mode thinking, there are so many possible choices for design to meet any one given requirement and following requirements does not guarantee an integrated overall solution.

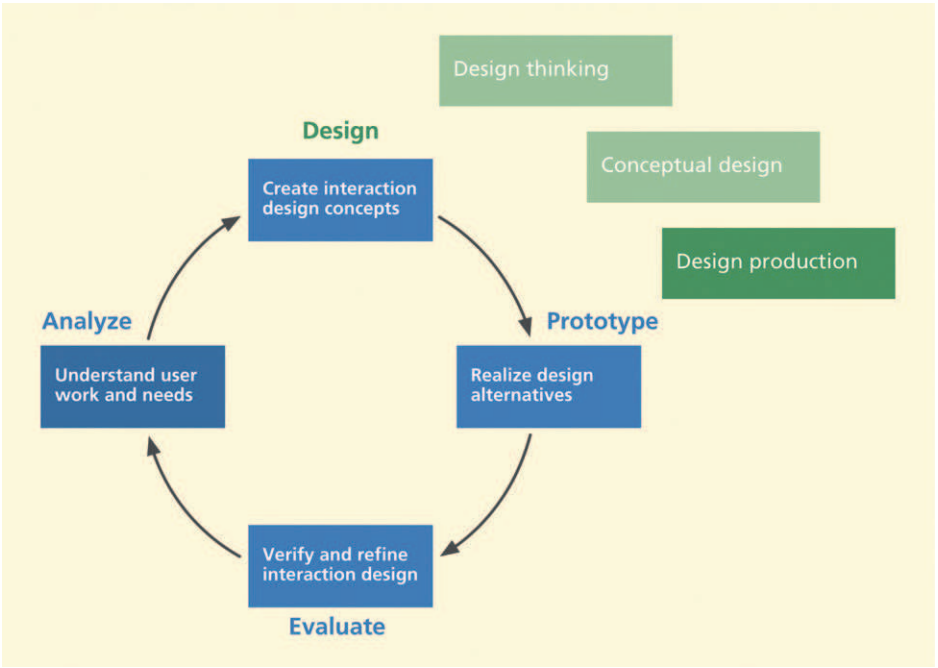


Figure 9-1
You are here; the third of three chapters on creating an interaction design in the context of the overall Wheel lifecycle template.

Beyer, Holtzblatt, and Wood (2005, p. 218) remind us that "The design isn't explicit in the data." "The data guides, constrains, and suggests directions" that design "can respond to." The requirements, whether in a requirements document or as an interpretation of the work activity affinity diagram (WAAD), offer a large inventory of things to be supported in the design.

9.2 MACRO VIEW OF LIFECYCLE ITERATIONS FOR DESIGN

In Figure 9-2 we show a "blow up" of how lifecycle iteration plays out on a macroscopic scale for the various types of design. Each type of design has its own iterative cycle with its own kind of prototype and evaluation. Among the very first to talk about iteration for interaction design were Buxton and Sniderman (1980).

The observant reader will note that the progressive series of iterative loops in Figure 9-2 can be thought of as a kind of spiral lifecycle concept. Each loop in turn addresses an increasing level of detail. For each different project context and each stage of progress within the project, you have to adjust the amount of and kind of design, prototyping, and evaluation to fit the situation in each of these incarnations of that lifecycle template.

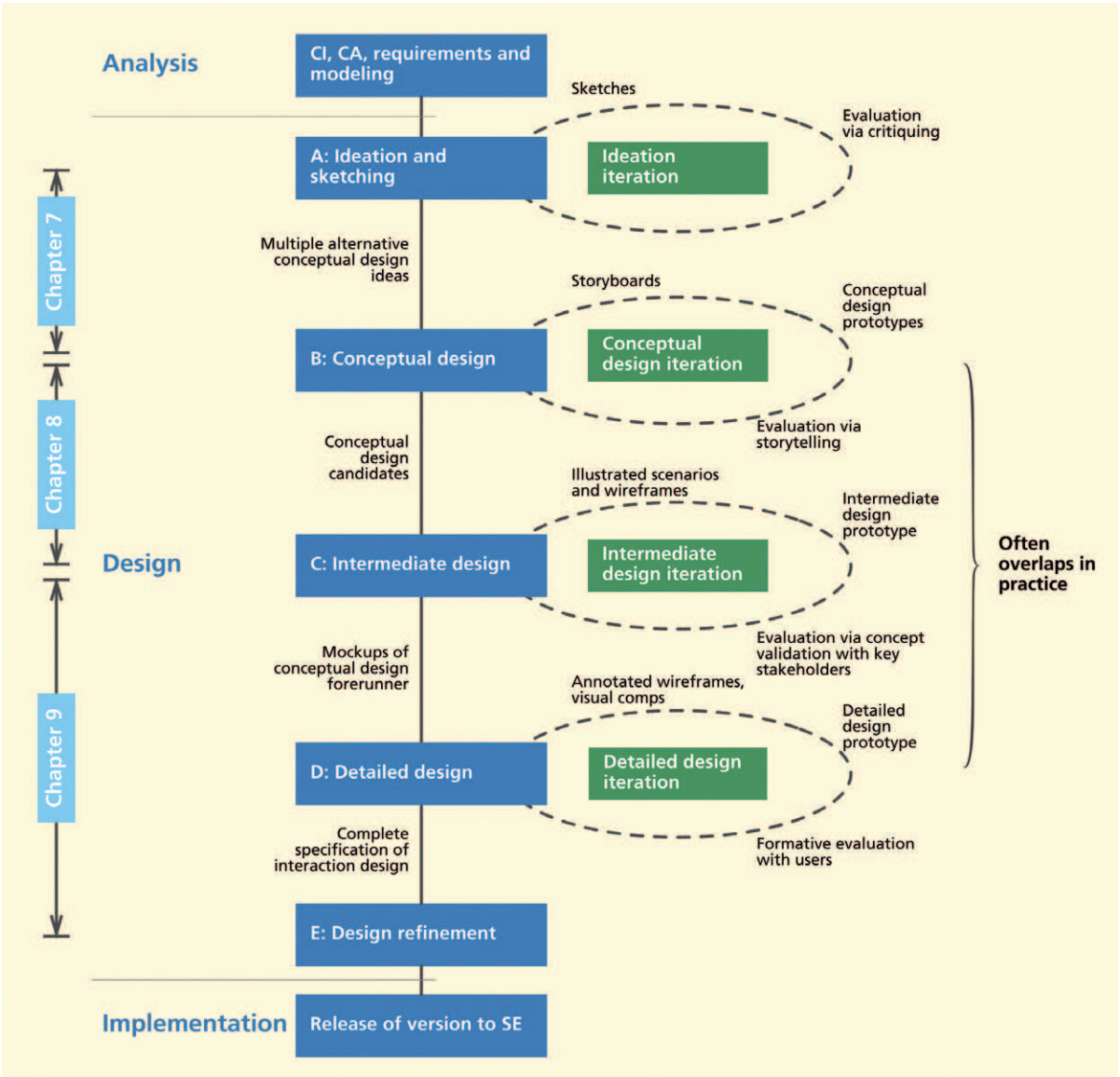


Figure 9-2
Macro view of lifecycle iterations in design.

9.2.1 Ideation Iteration

At "A" in Figure 9-2, iteration for ideation and sketching (Chapter 7) is a lightning-fast, loosely structured iteration for the purpose of exploring design ideas. The role of prototype is played by sketches, and the role of evaluation is carried out by brainstorming, discussion, and critiquing. Output is possibly multiple alternatives for conceptual designs, mostly in the form of annotated rough sketches.

Conceptual Design

A conceptual design is a theme, notion, or idea with the purpose of communicating a design vision about a system or product. It is the part of the system design that brings the designer's mental model to life.

Wireframe

A wireframe is a visual schematic, blueprint, or template of a screen or Web page design in an interaction design. It is a skeletal representation of screen (or page) layout of interaction objects such as tabs, menus, buttons, dialogue boxes, displays, and navigational elements. The focus of wireframes is on screen content and behavior but not graphical specifics such as fonts, colors, or graphics. Often the earliest way design ideas become tangible, wireframes are the basis for iterative rapid prototypes.

9.2.2 Conceptual Design Iteration

At "B" in Figure 9-2, iteration for conceptual design is to evaluate and compare possibly multiple design concepts and weigh concept feasibility. The type of prototype evolves with each successive iteration, roughly from paper prototype to low-fidelity wireframes and storyboards. The type of evaluation here is usually in the form of storytelling via storyboards to key stakeholders. The idea is to communicate how the broader design concepts help users in the envisioned work domain.

Depending on the project context, one or more of the design perspectives may be emphasized in the storyboards. This is usually the stage where key stakeholders such as users or their representatives, business, software engineering, and marketing must be heavily involved. You are planting the seeds for what the entire design will be for the system going forward.

9.2.3 Intermediate Design Iteration

At "C" in Figure 9-2, the purpose of intermediate design (coming up soon) iteration is to sort out possible multiple conceptual design candidates and to arrive at one intermediate design for layout and navigation. For example, for the Ticket Kiosk System, there are at least two conceptual design candidates in the interaction perspective. One is a traditional "drill-in" concept where users are shown available categories (e.g., movies, concerts, MU athletics) from which they choose one. Based on the choice on this first screen, the user is shown further options and details, navigating with a back button and/or "bread crumb" trail, if necessary, to come back to the category view. A second conceptual design is the one using the three-panel idea described in the previous chapter.

Intermediate prototypes might evolve from low-fidelity to high-fidelity or wireframes. Fully interactive high-fidelity mockups can be used as a vehicle to demonstrate leading conceptual design candidates to upper management stakeholders if you need this kind of communication at this stage. Using such wireframes or other types of prototypes, the candidate design concepts are validated and a conceptual design forerunner is selected.

9.2.4 Detailed Design Iteration

At "D" in Figure 9-2, iteration for detailed design is to decide screen design and layout details, including "visual comps" (coming up soon) of the "skin" for look and feel appearance. The prototypes might be detailed wireframes and/or high-fidelity interactive mockups. At this stage, the design will be fully

specified with complete descriptions of behavior, look and feel, and information on how all workflows, exception cases, and settings will be handled.

9.2.5 Design Refinement Iteration

At "E" in Figure 9-2, a prototype for refinement evaluation and iteration is usually medium to high fidelity and evaluation is either a rapid method (Chapter 13) or a full rigorous evaluation process (Chapters 12 and 14 through 18).

9.3 INTERMEDIATE DESIGN

For intermediate design, you will need the same team you have had since ideation and sketching, plus a visual designer if you do not already have one. Intermediate design starts with your conceptual design and moves forward with increasing detail and fidelity. The goal of intermediate design is to create a logical flow of intermediate-level navigational structure and screen designs. Even though we use the term screen here for ease of discussion, this is also applicable to other product designs where there are no explicit screens.

9.3.1 Unpacking the Conceptual Design: Strategies for Realization

At "C" in Figure 9-2, you are taking the concepts created in conceptual design, decomposing them into logical units, and expanding each unit into different possible design strategies (corresponding to different conceptual design candidates) for concept realization. Eventually you will decide on a design strategy, from which spring an iterated and evaluated intermediate prototype.

Information Object

An information object is an internally stored work object shared by users and the system. Information objects are often data entities central to work flow, being operated on by users; they are searched and browsed for, accessed and displayed, modified and manipulated, and stored back again.

9.3.2 Ground Your Design in Application Ontology with Information Objects

Per Johnson and Henderson (2002, p. 27), you should begin by thinking in terms of the ontological structure of the system, which will now be available in analyzed and structured contextual data. This starts with what we call information objects that we identified in modeling (Chapter 6).

As these information objects move within the envisioned flow model, they are accessed and manipulated by people in work roles. In a graphics-drawing application, for example, information objects might be rectangles, circles, and other graphical objects that are created, modified, and combined by users.

Identify relationships among the application objects—sometimes hierarchical, sometimes temporal, sometimes involving user workflow. With the

help of your physical model, cast your ontological net broadly enough to identify other kinds of related objects, for example, telephones and train tickets, and their physical manipulation as done in conjunction with system operation.

In design we also have to think about how users access information objects; from the user perspective, accessing usually means getting an object on the screen so that it can be operated on in some way. Then we have to think about what kinds of operations or manipulation will be performed.

For example, in the Ticket Kiosk System, events and tickets are important information objects. Start by thinking about how these can be represented in the design. What are the best design patterns to show an event? What are the design strategies to facilitate ways to manipulate them?

In your modeling you should have already identified information objects, their attributes, and relationships among them. In your conceptual design and later in intermediate design, you should already have decided how information objects will be represented in the user interaction design. Now you can decide how users get at, or access, these information objects.

Typically, because systems are too large and complex to show all information objects on the screen at once initially, how do your users call up a specific information object to operate on it? Think about information seeking, including browsing and searching.

Decide what operations users will carry out on your information objects. For example, a graphics package would have an operation to create a new rectangle object and operations to change its size, location, color, etc. Think about how users will invoke and perform those operations.

Add these new things to your storyboards. The design of information object operations goes hand in hand with design scenarios (Chapter 6), personas (Chapter 7), and storyboards (Chapter 8), which can add life to the static wireframe images of screens.

9.3.3 Illustrated Scenarios for Communicating Designs

One of the best ways to describe parts of your intermediate interaction design in a document is through illustrated scenarios, which combine the visual communication capability of storyboards and screen sketches with the capability of textual scenarios to communicate details. The result is an excellent vehicle for sharing and communicating designs to the rest of the team, and to management, marketing, and all other stakeholders.

Making illustrated scenarios is simple; just intersperse graphical storyboard frames and/or screen sketches as figures in the appropriate places to illustrate

the narrative text of a design scenario. The storyboards in initial illustrated scenarios can be sketches or early wireframes (coming up later).

9.3.4 Screen Layout and Navigational Structure

During this phase, all layout and navigation elements are fully fleshed out. Using sequences of wireframes, key workflows are represented while describing what happens when the user interacts with the different user interface objects in the design. It is not uncommon to have wireframe sets represent part of the workflow or each task sequence using click-through prototypes.

9.4 DETAILED DESIGN

At "D" in Figure 9-2, for detailed design you will need the same team you had for intermediate design, plus documentation and language experts, to make sure that the tone, vocabulary, and language are accurate, precise and consistent, both with itself and with terminology used in the domain.

9.4.1 Annotated Wireframes

To iterate and evaluate your detailed designs, refine your wireframes more completely by including all user interface objects and data elements, still represented abstractly but annotated with call-out text.

9.4.2 Visual Design and Visual Comps

As a parallel activity, a visual designer who has been involved in ideation, sketching, and conceptual design now produces what we call visual "comps," meaning variously comprehensive or composite layout (a term originating in the printing industry). All user interface elements are represented, now with a very specific and detailed graphical look and feel.

A visual comp is a pixel-perfect mockup of the graphical "skin," including objects, colors, sizes, shapes, fonts, spacing, and location, plus visual "assets" for user interface elements. An asset is a visual element along with all of its defining characteristics as expressed in style definitions such as cascading style sheets for a Website. The visual designer casts all of this to be consistent with company branding, style guides, and best practices in visual design.

Custom Style Guide

A custom style guide is a document that is fashioned and maintained by designers to capture and describe details of visual and other general design decisions that can be applied in multiple places. Its contents can be specific to one project or an umbrella guide across all projects on a given platform, or over a whole organization.

Exercise

See Exercise 9-1, Intermediate and Detailed Design for Your System

9.5 WIREFRAMES

In Figure 9-3 we show the path from ideation and sketching, task interaction models, and envisioned design scenarios to wireframes as representations of your designs for screen layout and navigational flow.

Along with ideation and sketching, task interaction models and design scenarios are the principal inputs to storytelling and communication of designs. As sequences of sketches, storyboards are a natural extension of sketching. Storyboards, like scenarios, represent only selected task threads. Fortunately, it is a short and natural step from storyboards to wireframes.

To be sure, nothing beats pencil/pen and paper or a whiteboard for the sketching needed in ideation (Chapter 7), but, at some point, when the design concept emerges from ideation, it must be communicated to others who pursue the rest of the lifecycle process. Wireframes have long been the choice in the field for documenting, communicating, and prototyping interaction designs.

9.5.1 What Are Wireframes?

Wireframes, a major bread-and-butter tool of interaction designers, are a form of prototype, popular in industry practice. Wireframes comprise lines and outlines (hence the name "wire frame") of boxes and other shapes to represent emerging interaction designs. They are schematic diagrams and "sketches" that define a Web page or screen content and navigational flow. They are used to illustrate high-level concepts, approximate visual layout, behavior, and sometimes even look and feel for an interaction design. Wireframes are embodiments of maps of screen or other state transitions during usage, depicting envisioned task flows in terms of user actions on user interface objects.

The drawing aspects of wireframes are often simple, offering mainly the use of rectangular objects that can be labeled, moved, and resized. Text and graphics

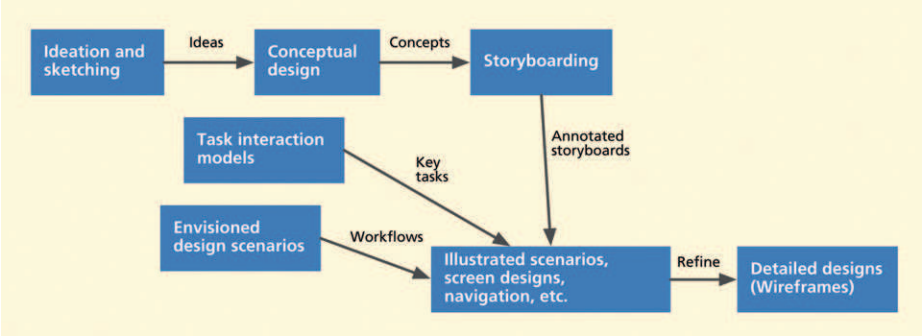


Figure 9-3
The path from ideation and sketching, task interaction models, and envisioned design scenarios to wireframes.

representing content and data in the design is placed in those objects. Drawing templates, or stencils, are used to provide quick means to represent the more common kinds of user interface objects (more on this in the following sections).

Wireframes are often deliberately unfinished looking; during early stages of design they may not even be to scale. They usually do not contain much visual content, such as finished graphics, colors, or font choices. The idea is to create design representations quickly and inexpensively by just drawing boxes, lines, and other shapes.

As an example of using wireframes to illustrate high-level conceptual designs, see Figure 9-4. The design concept depicted in this figure is comprised of a three-column pattern for a photo manipulation application. A primary navigation pane (the "nav bar") on the left-hand side is intended to show a list of all the user's photo collections. The center column is the main content display area for details, thumbnail images and individual photos, from the collection selected in the left pane.

The column on the right in Figure 9-4 is envisioned to show related contextual information for the selected collection. Note how a simple wireframe using just boxes, lines, and a little text can be effective in describing a broad

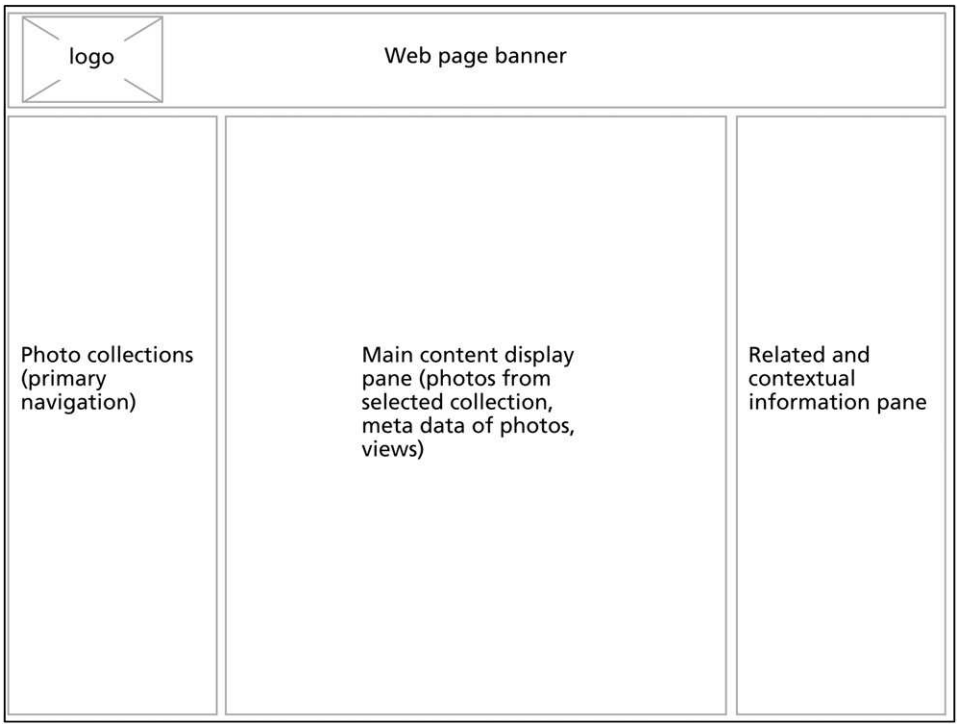


Figure 9-4

An example wireframe illustrating a high-level conceptual design.

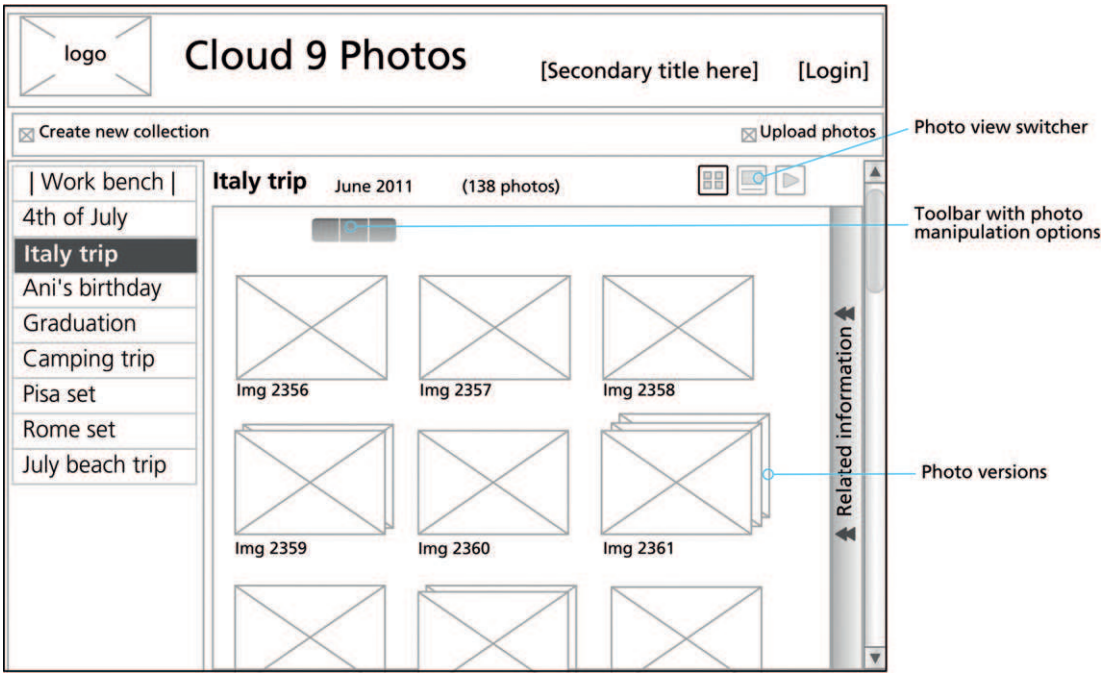


Figure 9-5

Further elaboration of the conceptual design and layout of Figure 9-4.

interaction conceptual design pattern. Often these kinds of patterns are explored during ideation and sketching, and selected sketches are translated into wireframes.

While wireframes can be used to illustrate high-level ideas, they are used more commonly to illustrate medium-fidelity interaction designs. For example, the idea of Figure 9-4 is elaborated further in Figure 9-5. The navigation bar in the left column now shows several picture collections and a default "work bench" where all uploaded images are collected. The selected item in this column, "Italy trip," is shown as the active collection using another box with the same label and a fill color of gray, for example, overlaid on the navigation bar. The center content area is also elaborated more using boxes and a few icons to show a scrollable grid of thumbnail images with some controls on the top right. Note how certain details pertaining to the different manipulation options are left incomplete while showing where they are located on the screen.

Wireframes can also be used to show behavior. For example, in Figure 9-6 we show what happens when a user clicks on the vertical "Related information" bar in Figure 9-5: a pane with contextual information for this collection (or individual photo) slides out. In Figure 9-7 we show a different view of the content

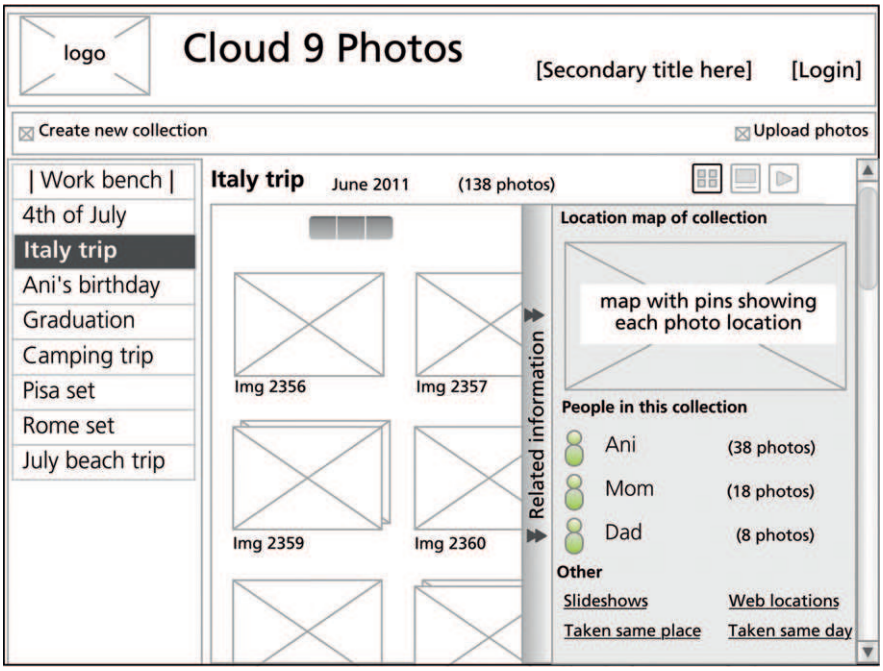


Figure 9-6

The display that results when a user clicks on the "Related information" bar.

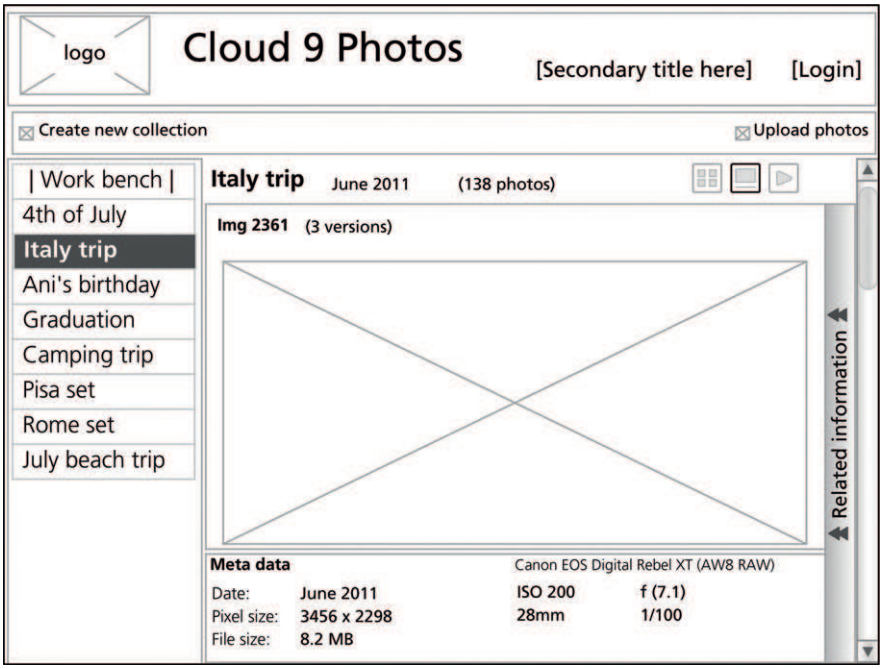


Figure 9-7

The display that results when a user clicks on the "One-up" view button.

pane, this time as a result of a user clicking on the "One-up" view switcher button in Figure 9-5 to see a single photo in the context pane. Double-clicking a thumbnail image will also expand that image into a one-up view to fill the content pane.

9.5.2 How Are Wireframes Used?

Wireframes are used as conversational props to discuss designs and design alternatives. They are effective tools to elicit feedback from potential users and other stakeholders. A designer can move through a deck of wireframes one slide at a time, simulating a potential scenario by pretending to click on interaction widgets on the screen. These page sequences can represent the flow of user activity within a scenario, but cannot show all possible navigational paths.

For example, if Figures 9-5, 9-6, and 9-7 are in a deck, a designer can narrate a design scenario where user actions cause the deck to progress through the corresponding images. Such wireframes can be used for rapid and early lab-based evaluation by printing and converting them into low-fidelity paper prototypes (Chapter 11). A rough low- to medium-fidelity prototype, using screens like the ones shown in Figures 9-5, 9-6, and 9-7, can also be used for design walkthroughs and expert evaluations. In the course of such an evaluation, the expert can extrapolate intermediate states between wireframes.

What we have described so far is easy to do with almost all wireframing tools. Most wireframing tools also provide hyperlinking capabilities to make the deck a click-through prototype. While this takes more effort to create, and even more to maintain as the deck changes, it provides a more realistic representation of the envisioned behavior of the design. However, the use of this kind of prototype in an evaluation might require elaborating all the states of the design in the workflow that is the focus of the evaluation.

Finally, after the design ideas are iterated and agreed upon by relevant stakeholders, wireframes can be used as interaction design specifications. When wireframes are used as inputs to design production, they are annotated with details to describe the different states of the design and widgets, including mouse-over states, keyboard inputs, and active focus states. Edge cases and transition effects are also described. The goal here is completeness, to enable a developer to implement the designs without the need for any interpretation. Such specifications are usually accompanied by high-fidelity visual comps, discussed previously in this chapter.

9.5.3 How to Build Wireframes?

Wireframes can be built using any drawing or word processing software package that supports creating and manipulating shapes, such as iWork Pages, Keynote, Microsoft PowerPoint, or Word. While such applications suffice for simple wireframing, we recommend tools designed specifically for this purpose, such as OmniGraffle (for Mac), Microsoft Visio (for PC), and Adobe InDesign.

Many tools and templates for making wireframes are used in combination—truly an invent-as-you-go approach serving the specific needs of prototyping. For example, some tools are available to combine the generic-looking placeholders in wireframes with more detailed mockups of some screens or parts of screens. In essence they allow you to add color, graphics, and real fonts, as well as representations of real content, to the wireframe scaffolding structure.

In early stages of design, during ideation and sketching, you started with thinking about the high-level conceptual design. It makes sense to start with that here, too, first by wireframing the design concept and then by going top down to address major parts of the concept. Identify the interaction conceptual design using boxes with labels, as shown in Figure 9-4.

Take each box and start fleshing out the design details. What are the different kinds of interaction needed to support each part of the design, and what kinds of widgets work best in each case? What are the best ways to lay them out? Think about relationships among the widgets and any data that need to go with them. Leverage design patterns, metaphors, and other ideas and concepts from the work domain ontology. Do not spend too much time with exact locations of these widgets or on their alignment yet. Such refinement will come in later iterations after all the key elements of the design are represented.

As you flesh out all the major areas in the design, be mindful of the information architecture on the screen. Make sure the wireframes convey that inherent information architecture. For example, do elements on the screen follow a logical information hierarchy? Are related elements on the screen positioned in such a way that those relationships are evident? Are content areas indented appropriately? Are margins and indents communicating the hierarchy of the content in the screen?

Next it is time to think about sequencing. If you are representing a workflow, start with the "wake-up" state for that workflow. Then make a wireframe representing the next state, for example, to show the result of a user action such as clicking on a button. In Figure 9-6 we showed what happens when a user clicks

on the "Related information" expander widget. In Figure 9-7 we showed what happens if the user clicks on the "One-up" view switcher button.

Once you create the key screens to depict the workflow, it is time to review and refine each screen. Start by specifying all the options that go on the screen (even those not related to this workflow). For example, if you have a toolbar, what are all the options that go into that toolbar? What are all the buttons, view switchers, window controllers (e.g., scrollbars), and so on that need to go on the screen? At this time you are looking at scalability of your design. Is the design pattern and layout still working after you add all the widgets that need to go on this screen?

Think of cases when the windows or other container elements such as navigation bars in the design are resized or when different data elements that need to be supported are larger than shown in the wireframe. For example, in Figures 9-5 and 9-6, what must happen if the number of photo collections is greater than what fits in the default size of that container? Should the entire page scroll or should new scrollbars appear on the left-hand navigation bar alone? How about situations where the number of people identified in a collection are large? Should we show the first few (perhaps ones with most number of associated photos) with a "more" option, should we use an independent scrollbar for that pane, or should we scroll the entire page? You may want to make wireframes for such edge cases; remember they are less expensive and easier to do using boxes and lines than in code.

As you iterate your wireframes, refine them further, increasing the fidelity of the deck. Think about proportions, alignments, spacing, and so on for all the widgets. Refine the wording and language aspects of the design. Get the wireframe as close to the envisioned design as possible within the constraints of using boxes and lines.

9.5.4 Hints and Tips for Wireframing

Because the point of wireframing is to make quick prototypes for exploring design ideas, one of the most important things to remember about wireframing is modularity. Just as in paper prototyping, you want to be able to create multiple design representations quickly.

Being modular means not having too many concepts or details "hard coded" in any one wireframe. Build up concepts and details using "layers." Most good wireframing tools provide support for layers that can be used to abstract related design elements into reusable groups. Use a separate layer for each repeating set of widgets on the screen. For example, the container "window" of the

application with its different controls can be specified once as a layer and this layer can be reused in all subsequent screens that use that window control.

Similarly, if there is a navigation area that is not going to change in this wireframe deck, for example, the left-hand collections pane in Figure 9-5, use one shared layer for that. Layers can be stacked upon one another to construct a slide. This stacking also provides support for ordering in the *Z* axis to show overlapping widgets. Selection highlights, for example, showing that "Italy trip" is the currently selected collection in Figure 9-5, can also created using a separate "highlight" layer.

Another tip for efficient wireframing is to use stencils, templates, and libraries of widgets. Good wireframing tools often have a strong community following of users who share wireframing stencils and libraries for most popular domains—for example, for interaction design—and platforms—for example, Web, Apple iOS, Google's Android, Microsoft's Windows, and Apple's Macintosh. Using these libraries, wireframing becomes as easy as dragging and dropping different widgets onto layers on a canvas.

Create your own stencil if your design is geared toward a proprietary platform or system. Start with your organization's style guide and build a library of all common design patterns and elements. Apart from efficiency, stencils and libraries afford consistency in wireframing.

Some advanced wireframing tools even provide support for shared objects in a library. When these objects are modified, it is possible to automatically update all instances of those objects in all linked wireframe decks. This makes maintenance and updates to wireframes easier.

Sketchy wireframes

Sometimes, when using wireframes to elicit feedback from users, if you want to convey the impression that the design is still amenable to changes, make wireframes look like sketches. We know from Buxton (2007a) that the style or "language" of a sketch should not convey the perception that it is more developed than it really is. Straight lines and coloring within the lines give the false impression that the design is almost finished and, therefore, constructive criticism and new ideas are no longer appropriate.

However, conventional drawing tools, such as Microsoft Visio, Adobe Illustrator, OmniGraffle, and Adobe inDesign, produce rigid, computer-drawn boxes, lines, and text. In response, "There is a growing popularity toward something in the middle: Computer-based sketchy wireframes. These allow computer wireframes to look more like quick, hand-drawn sketches while retaining the reusability and polish that we expect from digital artifacts" (Travis, 2009).

CHAPTER

UX Goals, Metrics, and Targets

10

Objectives

After reading this chapter, you will:

1. Understand the concepts of UX goals, metrics, and targets
2. Appreciate the need for setting UX target values for the envisioned system
3. Understand the influence of user classes, business goals, and UX goals on UX targets
4. Be able to create UX target tables, including identifying measuring instruments and setting target values
5. Know how UX targets help manage the UX lifecycle process

10.1 INTRODUCTION

10.1.1 You Are Here

We are making splendid progress in moving through the Wheel UX lifecycle template. In this chapter we establish operational targets for user experience to assess the level of success in your designs so that you know when you can move on to the next iteration. UX goals, metrics, and targets help you plan for evaluation that will successfully reveal the user performance and emotional satisfaction bottlenecks. Because UX goals, metrics, and targets are used to guide much of the process from analysis through evaluation, we show it as an arc around the entire lifecycle template, as you can see in Figure 10-1.

10.1.2 Project Context for UX Metrics and Targets

In early stages, evaluation usually focuses on qualitative data for finding UX problems. In these early evaluations the absence of quantitative data precludes the use of UX metrics and targets. But you may still want to establish them at this point if you intend to use them in later evaluations.

However, there is another need why you might forego UX metrics and targets. In most practical contexts, specifying UX metrics and targets and following up with

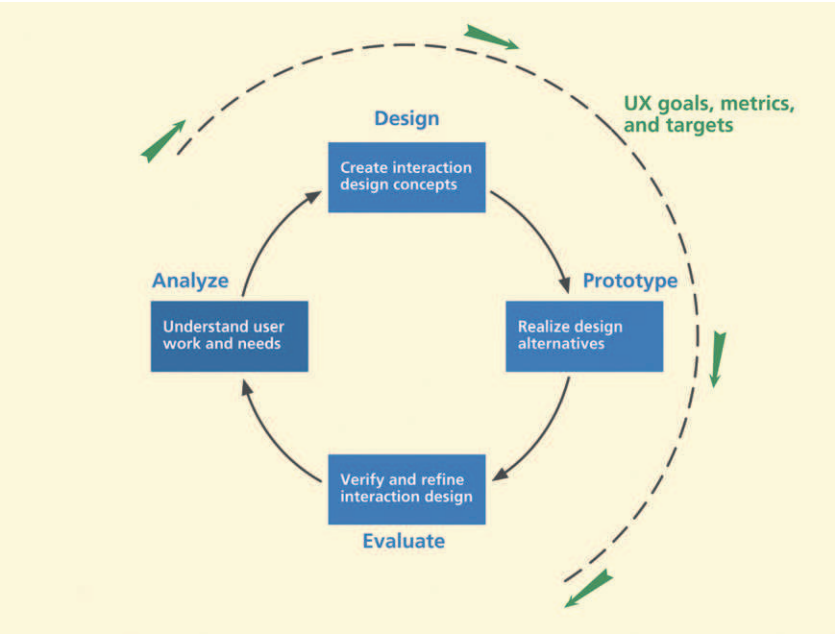


Figure 10-1

You are here; the chapter on UX goals, metrics, and targets in the context of the overall Wheel lifecycle template.

them may be too expensive. This level of completeness is only possible in a few organizations where there are established UX resources. In most places, one round of evaluation is all one gets. Also, as designers, we can know which parts of the design need further investigation just by looking at the results of the first round of evaluation. In such cases, quantitative UX metrics and targets may not be useful but benchmark tasks are still essential as vehicles for driving evaluation.

Regardless, the trend in the UX field is moving away from a focus on user performance and more toward user satisfaction and enjoyment. We include the full treatment of UX goals, metrics, and targets here and quantitative data collection and analysis in the later UX evaluation chapters for completeness and because some readers and practitioners still want coverage of the topic.

In any case, we find that this pivotal interaction design process activity of specifying UX goals, metrics, and targets is often overlooked, either because of lack of knowledge or because of lack of time. Sometimes this can be unfortunate because it can diminish the potential of what can be accomplished with the resources you will be putting into user experience evaluation. This chapter will help you avoid that pitfall by showing you techniques for specifying UX goals, metrics, and targets.

Fortunately, creating UX metrics and targets, after a little practice, does not take much time. You will then have specific quantified UX goals against which to test rather than just waiting to see what happens when you put users in front of your interaction design. Because UX metrics and targets provide feasible objectives for formative evaluation efforts, the results can help you pinpoint where to focus on redesign most profitably.

And, finally, UX goals, metrics, and targets offer a way to help manage the lifecycle by defining a quantifiable end to what can otherwise seem like endless iteration. Of course, designers and managers can run out of time, money, and

Benchmark Task

A benchmark task is a description of a task performed by a participant in formative evaluation so that UX measures such as time-on-task and error rates can be obtained and compared to a baseline value across the performances of multiple participants.

patience before they meet their UX targets—sometimes after just one round of evaluation—but at least then they know where things stand.

10.1.3 Roots for UX Metrics and Targets

The concept of formal UX measurement specifications in tabular form, with various metrics operationally defining success, was originally developed by Gilb (1987). The focus of Gilb's work was on using measurements in managing software development resources. Bennett (1984) adapted this approach to usability specifications as a technique for setting planned usability levels and managing the process to meet those levels.

These ideas were integrated into usability engineering practice by Good et al. (1986) and further refined by Whiteside, Bennett, and Holtzblatt (1988). Usability engineering, as defined by Good et al. (1986), is a process through which quantitative usability characteristics are specified early and measured throughout the lifecycle process.

Carroll and Rosson (1985) also stressed the need for quantifiable usability specifications, associated with appropriate benchmark tasks, in iterative refinement of user interaction designs. And now we have extended the concept to UX targets. Without measurable targets, it is difficult to determine, at least quantitatively, whether the interaction design for a system or product is meeting your UX goals.

10.2 UX GOALS

UX goals are high-level objectives for an interaction design, stated in terms of anticipated user experience. UX goals can be driven by business goals and reflect real use of a product and identify what is important to an organization, its customers, and its users. They are expressed as desired effects to be experienced in usage by users of features in the design and they translate into a set of UX measures. A UX measure is a usage attribute to be assessed in evaluating a UX goal.

You will extract your UX goals from user concerns captured in work activity notes, the flow model, social models, and work objectives, some of which will be market driven, reflecting competitive imperatives for the product. User experience goals can be stated for all users in general or in terms of a specific work role or user class or for specific kinds of tasks.

Examples of user experience goals include ease-of-use, power performance for experts, avoiding errors for intermittent users, safety for life-critical systems, high customer satisfaction, walk-up-and-use learnability for new users, and so on.

Example: User Experience Goals for Ticket Kiosk System

We can define the primary high-level UX goals for the ticket buyer to include:

- Fast and easy walk-up-and-use user experience, with absolutely no user training
- Fast learning so new user performance (after limited experience) is on par with that of an experienced user [from AB-4-8]
- High customer satisfaction leading to high rate of repeat customers [from BC-6-16]

Some other possibilities:

- High learnability for more advanced tasks [from BB-1-5]
- Draw, engagement, attraction
- Low error rate for completing transactions correctly, especially in the interaction for payment [from CG-13-17]

Exercise

See Exercise 10-1, Identifying User Experience Goals for Your System

10.3 UX TARGET TABLES

Through years of working with real-world UX practitioners and doing our own user experience evaluations, we have refined the concept of a UX target table, in the form shown in Table 10-1, from the original conception of a usability specification table, as presented by Whiteside, Bennett, and Holtzblatt (1988). A spreadsheet is an obvious way to implement these tables.

For convenience, one row in the table is called a "UX target." The first three columns are for the work role and related user class to which this UX target applies, the associated UX goal, and the UX measure. The three go together because each UX measure is aimed at supporting a UX goal and is specified with respect to a work role and user class combination. Next, we will see where you get the information for these three columns.

As a running example to illustrate the use of each column in the UX target table, we will progressively set some UX targets for the Ticket Kiosk System.

Table 10-1

Our UX target table, as evolved from the Whiteside, Bennett, and Holtzblatt (1988) usability specification table

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results

10.4 WORK ROLES, USER CLASSES, AND UX GOALS

Because UX targets are aimed at specific work roles, we label each UX target by work role. Recall that different work roles in the user models perform different task sets.

So the key task sets for a given work role will have associated usage scenarios, which will inform benchmark task descriptions we create as measuring instruments to go with UX targets. Within a given work role, different user classes will generally be expected to perform to different standards, that is, at different target levels.

Measuring Instrument

A measuring instrument is the means for providing values for a particular UX measure; it is the vehicle through which values are generated and measured. A typical measuring instrument for generating objective UX data is a benchmark task—for example, user performance of a task gives time and error data—while a typical measuring instrument for generating subjective UX data is a questionnaire.

Example: A Work Role, User Class, and UX Goal for the Ticket Kiosk System

In Table 10-1, we see that the first values to enter for a UX target are work role, a corresponding user class, and related UX goal. As we saw earlier, user class definitions can be based on, among other things, level of expertise, disabilities and limitations, and other demographics.

For the Ticket Kiosk System, we are focusing primarily on the ticket buyer. For this work role, user classes include a casual town resident user from Middleburg and a student user from the Middleburg University. In this example, we feature the casual town user.

Translating the goal of "fast-and-easy walk-up-and-use user experience" into a UX target table entry is straightforward. This goal refers to the ability of a typical occasional user to do at least the basic tasks on the first try, certainly without training or manuals. Typing them in, we see the beginnings of a UX target in the first row of Table 10-2.

Table 10-2

Choosing a work role, user class, and UX goal for a UX target

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user						

10.5 UX MEASURES

Within a UX target, the UX measure is the general user experience characteristic to be measured with respect to usage of your interaction design. The choice of UX measure implies something about which types of measuring instruments and UX metrics are appropriate.

UX targets are based on quantitative data—both objective data, such as observable user performance, and subjective data, such as user opinion and satisfaction.

Some common UX measures that can be paired with quantitative metrics include:

- Objective UX measures (directly measurable by evaluators)
  - Initial performance
  - Long-term performance (longitudinal, experienced, steady state)
  - Learnability
  - Retainability
  - Advanced feature usage
- Subjective UX measures (based on user opinions)
  - First impression (initial opinion, initial satisfaction)
  - Long-term (longitudinal) user satisfaction

Initial performance refers to a user's performance during the very first use (somewhere between the first few minutes and the first few hours, depending on the complexity of the system). Long-term performance typically refers to performance during more constant use over a longer period of time (fairly regular use over several weeks, perhaps). Long-term usage usually implies a steady-state learning plateau by the user; the user has become familiar with the system and is no longer constantly in a learning state.

Initial performance is a key UX measure because any user of a system must, at some point, use it for the first time. Learnability and retainability refer, respectively, to how quickly and easily users can learn to use a system and how well they retain what they have learned over some period of time.

Advanced feature usage is a UX measure that helps determine user experience of more complicated functions of a system. The user's initial opinion of the system can be captured by a first impression UX measure, whereas long-term user satisfaction refers, as the term implies, to the user's opinion after using the system for some greater period of time, after some allowance for learning.

Initial performance and first impression are appropriate UX measures for virtually every interaction design. Other UX measures often play support roles to address more specialized UX needs. Conflicts among UX measures are not unheard of. For example, you may need both good learnability and good expert performance. In the design, those requirements can work against each other. This, however, just reflects a normal kind of design trade-off. UX targets based on the two different UX measures imply user performance requirements pulling in two different directions, forcing the designers to stretch the design and face the trade-off honestly.

Example: UX Measures for the Ticket Kiosk System

For the walk-up ease-of-use goal of our casual new user, let us start simply with just two UX measures: initial performance and first impression. Each UX measure will appear in a separate UX target in the UX target table, with the user class of the work role and UX goal repeated, as in Table 10-3.

10.6 MEASURING INSTRUMENTS

Within a UX target, the measuring instrument is a description of the method for providing values for the particular UX measure. The measuring instrument is how data are generated; it is the vehicle through which values are measured for the UX measure.

Although you can get creative in choosing your measuring instruments, objective measures are commonly associated with a benchmark task—for example, a time-on-task measure as timed on a stopwatch, or an error rate measure made by counting user errors—and subjective measures are commonly associated with a user questionnaire—for example, the average user rating-scale scores for a specific set of questions.

Table 10-3

Choosing initial performance and first impression as UX measures

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance					
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression					

For example, we will see that the objective "initial user performance" UX measure in the UX target table for the Ticket Kiosk System is associated with a benchmark task and the "first impression" UX measure is associated with a questionnaire. Both subjective and objective measures and data can be important for establishing and evaluating user experience coming from a design.

10.6.1 Benchmark Tasks

According to Reference.com, the term "benchmark" originates in surveying, referring to:

> Chiseled horizontal marks that surveyors made in stone structures, into which an angle-iron could be placed to form a "bench" for a leveling rod, thus ensuring that a leveling rod could be accurately repositioned in the same place in future. These marks were usually indicated with a chiseled arrow below the horizontal line.

As a measuring instrument for an objective UX measure, a benchmark task is a representative task that you will have user participants perform in evaluation where you can observe their performance and behavior and take qualitative data (on observations of critical incidents and user experience problems) and quantitative data (user performance data to compare with UX targets). As such, a benchmark task is a "standardized" task that can be used to compare (as an engineering comparison, not a rigorous scientific comparison) performance among different users and across different design versions.

Address designer questions with benchmark tasks and UX targets

As designers work on interaction designs, questions arise constantly. Sometimes the design team simply cannot decide an issue for themselves and they defer it to UX testing ("let the users decide"). Perhaps the team does not agree on a way to treat one design feature, but they have to pick something in order to move forward.

Maybe you do agree on the design for a feature but you are very curious about how it will play out with real users. Perchance you do not believe an input you got in your requirements from contextual analysis but you used it, anyway, and now you want to see if it pans out in the design.

We have suggested that you keep a list of design questions as they came up in design activities. Now they play a role in setting benchmark tasks to get

feedback from users regarding these questions. Benchmark tasks based on designer issues are often the only way this kind of issue will get considered in evaluation.

Selecting benchmark tasks

In general, of course, the benchmark tasks you choose as measuring instruments should closely represent tasks real users will perform in a real work context. Pick tasks where you think or know the design has weaknesses. Avoiding such tasks violates the spirit of UX targets and user experience evaluation; it is about finding user experience problems so that you can fix them, not about proving you are the best designer. If you think of UX targets as a measure of how good you are as a designer, you will have a conflict of interest because you are setting your own evaluation criteria. That is not the point of UX targets at all.

Here are some guidelines for creating effective benchmark tasks.

Create benchmark tasks for a representative spectrum of user tasks. Choose realistic tasks intended to be used by each user class of a work role across the system. To get the best coverage for your evaluation investment, your choices should represent the cross section of real tasks with respect to frequency of performance and criticality to goals of the users of the envisioned product. Benchmark tasks are also selected to evaluate new features, "edge cases" (usage at extreme conditions), and business-critical and mission-critical tasks. While some of these tasks may not be performed frequently, getting them wrong could cause serious consequences.

Start with short and easy tasks and then increase difficulty progressively. Because your benchmark tasks will be faced by participant users in a sequence, you should consider their presentation order. In most cases, start with relatively easy ones to get users accustomed to the design and feeling comfortable in their role as evaluators. After building user confidence and engagement, especially with the tasks for the "initial performance" UX measure, you can introduce more features, more breadth, variety, complexity, and higher levels of difficulty.

In some cases, you might have your user participants repeat a benchmark task, only using a different task path, to see how users get around in multiple ways. The more advanced benchmark tasks are also a place to try your creativity by introducing intervening circumstances. For example, you might lead the user

down a path and then say "At this point, you change your mind and want to do such and such, departing from where you are now."

For our ticket kiosk system, maybe start with finding a movie that is currently playing. Then follow with searching for and reserving tickets for a movie that is to be showing 20 days from now and then go to more complex tasks such as purchasing concert tickets with seat and ticket type selection.

Include some navigation where appropriate. In real usage, because users usually have to navigate to get to where they will do the operations specific to performing a task, you want to include the need for this navigation even in your earliest benchmark tasks. It tests their knowledge of the fact that they do need to go elsewhere, where they need to go, and how to get there.

Avoid large amounts of typing (unless typing skill is being evaluated). Avoid anything in your benchmark task descriptions that causes large user performance variation not related to user experience in the design. For example, large amounts of typing within a benchmark task can cause large variations in user performance, but the variations will be based on differences in typing skills and can obscure performance differences due to user experience or usability issues.

Match the benchmark task to the UX measure. Obviously, if the UX measure is "initial user performance," the task should be among those a first-time user realistically would face. If the UX measure is about advanced feature usage, then, of course, the task should involve use of that feature to match this requirement. If the UX measure is "long-term usage," then the benchmark task should be faced by the user after considerable practice with the system. For a UX measure of "learnability," a set of benchmark tasks of increasing complexity might be appropriate.

Adapt scenarios already developed for design. Design scenarios clearly represent important tasks to evaluate because they have already been selected as key tasks in the design. However, you *must* remember to remove information about how to perform the tasks, which is usually abundant in a scenario. See guideline "Tell the user *what* task to do, but not *how* to do it" in the next section for more discussion.

Use tasks in realistic combinations to evaluate task flow. To measure user performance related to task flow, use combinations of tasks such as those that will occur together frequently. In these cases, you should set UX targets

for such combinations because difficulties related to user experience that appear during performance of the combined tasks can be different than for the same tasks performed separately. For example, in the Ticket Kiosk System, you may wish to measure user performance on the task thread of searching for an event and then buying tickets for that event.

As another example, a benchmark task might require users to buy four tickets for a concert under a total of $200 while showing tickets in this price range for the upcoming few days as sold out. This would force users to perform the task of searching through other future concert days, looking for the first available day with tickets in this price range.

Do not forget to evaluate with your power users. Often user experience for power users is addressed inadequately in product testing (Karn, Perry, & Krolczyk, 1997). Do your product business and UX goals include power use by a trained user population? Do they require support for rapid repetition of tasks, complex and possibly very long tasks? Does their need for productivity demand shortcuts and direct commands over interactive hand-holding?

If any of these are true, you must include benchmark tasks that match this kind of skilled and demanding power use. And, of course, these benchmark tasks must be used as the measuring instrument in UX targets that match up with the corresponding user classes and UX goals.

To evaluate error recovery, a benchmark task can begin in an error state. Effective error recovery is a kind of "feature" that designers and evaluators can easily forget to include. Yet no interaction design can guarantee error-free usage, and trying to recover from errors is something most users are familiar with and can relate to. A "forgiving" design will allow users to recover from errors relatively effortlessly. This ability is definitely an aspect of your design that should be evaluated by one or more benchmark tasks.

Consider tasks to evaluate performance in "degraded modes" due to partial equipment failure. In large interconnected, networked systems such as military systems or large commercial banking systems, especially involving multiple kinds of hardware, subsystems can go down. When this happens, will your part of the system give up and die or can it at least continue some of its intended functionality and give partial service in a "degraded mode?" If your application fits this description, you should include benchmark tasks to evaluate the user's perspective of this ability accordingly.

Do not try to make a benchmark task for everything. Evaluation driven by UX targets is only an engineering sampling process. It will not be possible to establish UX targets for all possible classes of users doing all possible tasks. It is often stated that about 20% of the tasks in an interactive system account for 80% of the usage and vice versa. While these figures are obviously folkloric guesses, they carry a grain of truth to guide in targeting users and tasks in establishing UX targets.

Constructing benchmark task content

Here we list a number of tips and hints to consider when creating benchmark task content.

Remove any ambiguities with clear, precise, specific, and repeatable instructions. Unless resolving ambiguity is what we want users to do as part of the task, we must make the instructions in benchmark task descriptions clear and not confusing. Unambiguous benchmark tasks are necessary for consistent results; we want differences in user performance to be due to differences in users or differences in designs but usually not due to different interpretations of the same benchmark task.

As a subtle example, consider this "add appointment" benchmark task for the "initial performance" UX measure for an interdepartmental event scheduling system. Schedule a meeting with Dr. Ehrich for a month from today at 10 AM in 133 McBryde Hall concerning the HCI research project.

For some users, the phrase "1 month from today" can be ambiguous. Why? It can mean, for example, on the same date next month or it can mean exactly 4 weeks from now, putting it on the same day of the week. If that difference in meaning can make a difference in user task performance, you need to make the wording more specific to the intended meaning.

You also want to make your benchmark tasks specific so that participants do not get sidetracked on irrelevant details during testing. If, for example, a "find event" benchmark task is stated simply as "Find an entertainment event for sometime next week," some participants might make it a long, elaborate task, searching around for some "best" combination of event type and date, whereas others would do the minimum and take the first event they see on the screen. To mitigate such differences, add specific information about event selection criteria.

Tell the user *what* task to do, but not *how* to do it. This guideline is very important; the success of user experience evaluation based on this task will depend on it. Sometimes we find students in early evaluation exercises

presenting users with task instructions that spell out a series of steps to perform. They should not be surprised when the evaluation session leads to uninteresting results.

The users are just giving a rote performance of the steps as they read them from the benchmark task description. If you wish to test whether your interaction design helps users discover how to do a given task on their own, you must avoid giving any information about *how* to do it. Just tell them *what* task to do and let them figure out how.

Example (to do): "Buy two student tickets for available adjacent seats as close to the stage as possible for the upcoming Ben King concert and pay with a credit card."

Example (not to do): "Click on the Special Events button on the home screen; then select More at the bottom of the screen. Select the Ben King concert and click on Seating Options. . . ."

Example (not to do): "Starting at the Main Menu, go to the Music Menu and set it as a Bookmark. Then go back to the Main Menu and use the Bookmark feature to jump back to the Music Menu."

Do not use words in benchmark tasks that appear specifically in the interaction design. In your benchmark task descriptions, you must avoid using any words that appear in menu headings, menu choices, button labels, icon pop-ups, or any place in the interaction design itself. For example, do not say "Find the first event (that has such and such a characteristic)" when there is a button in the interaction design labeled "Find." Instead, you should use words such as "Look for . . ." or "Locate . . ."

Otherwise it is very convenient for your users to use a button labeled "Find" when they are told to "Find" something. It does not require them to think and, therefore, does not evaluate whether the design would have helped them find the right button on their own in the course of real usage.

Use work context and usage-centered wording, not system-oriented wording. Because benchmark task descriptions are, in fact, descriptions of user tasks and not system functionality, you should use usage-centered words from the user's work context and not system-centered wording. For example, "Find information about xyz" is better than "Submit query about xyz." The former is task oriented; the latter is more about a system view of the task.

Have clear start and end points for timing. In your own mind, be sure that you have clearly observable and distinguishable start and end points for each benchmark task and make sure you word the benchmark task description

to use these end points effectively. These will ensure your ability to measure the time on task accurately, for example.

At evaluation time, not only must the evaluators know for sure when the task is completed, but *the participant must know when the task is completed.* For purposes of evaluation, the task cannot be considered completed until the user experiences closure.

The evaluator must also know when the user knows that the task has been completed. Do not depend on the user to say when the task is done, even if you explicitly ask for that in the benchmark task description or user instructions. Therefore, rather than ending task performance with a mental or sensory state (i.e., the user knowing or seeing something), it is better to incorporate a user action confirming the end of the task, as in the (to do) examples that follow.

Example (not to do): "Find out how to set the orientation of the printer paper to "landscape." Completion of this task depends on the user knowing something and that is not a directly observable state. Instead, you could have the user actually set the paper orientation; this is something you can observe directly.

Example (not to do): "View next week's events." Completion of this task depends on the user seeing something, an action that you may not be able to confirm. Perhaps you could have the user view and read aloud the contents of the first music event next week. Then you know whether and when the user has seen the correct event.

Example (to do): "Find next week's music event featuring Rachel Snow and add it to the shopping cart."

Example (to do): Or, to include knowing or learning how to select seats, "Find the closest available seat to the stage and add to shopping cart."

Example (to do): "Find the local weather forecast for tomorrow and read it aloud."

Keep some mystery in it for the user. Do not always be too specific about what the users will see or the parameters they will encounter. Remember that real first-time users will approach your application without necessarily knowing how it works. Sometimes try to use benchmark tasks that give approximate values for some parameters to look for, letting the rest be up to the user. You can still create a prototype in such a way that there is only one possible "solution" to this task if you want to avoid different users in the evaluation ending in a different state in the system.

Example (to do): "Purchase two movie tickets to *Bee Movie* within 1.5 hours of the current time and showing at a theatre within 5 miles of this kiosk location."

Annotate situations where evaluators must ensure pre-conditions for running benchmark tasks. Suppose you write this benchmark task: "Your dog, Mutt, has just eaten your favorite book and you have decided that he is not worth spending money on. Delete your appointment with the vet for Mutt's annual checkup from your calendar."

Every time a user performs this task during evaluation, the evaluator must be sure to have an existing appointment already in your prototype calendar so that each user can find it and delete it. You must attach a note in the form of rubrics (next point later) to this benchmark task to that effect—a note that will be read and followed much later, in the evaluation activity.

Ecological Validity

Ecological validity refers to the realism with which a design of evaluation setup matches the user's real work context. It is about how accurately the design or evaluation reflects the relevant characteristics of the ecology of interaction, i.e., its context in the world or its environment.

Use "rubrics" for special instructions to evaluators. When necessary or useful, add a "rubrics" section to your benchmark task descriptions as special instructions to evaluators, not to be given to participants in evaluation sessions. Use these rubrics to communicate a heads-up about anything that needs to be done or set up in advance to establish task preconditions, such as an existing event in the kiosk system, work context for ecological validity, or a particular starting state for a task.

Benchmark tasks for addressing designer questions are especially good candidates for rubrics. In a note accompanying your benchmark task you can alert evaluators to watch for user performance or behavior that might shed light on these specific designer questions.

Put each benchmark task on a separate sheet of paper. Yes, we want to save trees but, in this case, it is necessary to present the benchmark tasks to the participant only one at a time. Otherwise, the participant will surely read ahead, if only out of curiosity, and can become distracted from the task at hand.

If a task has a surprise step, such as a midtask change of intention, that step should be on a separate piece of paper, not shown to the participant initially. To save trees you can cut (with scissors) a list of benchmark tasks so that only one task appears on one piece of paper.

Write a "task script" for each benchmark task. You should write a "task script" describing the steps of a representative or typical way to do the task and include it in the benchmark task document "package." This is just for use by the evaluator and is definitely not given to the participant. The evaluator may not have been a member of the design team and initially may not be too familiar with how to perform the benchmark tasks, and it helps the evaluator to be able to

anticipate a possible task performance path. This is especially useful in cases where the participant cannot determine a way to do the task; then, the evaluation facilitator knows at least one way.

Example: Benchmark Tasks as Measuring Instruments for the Ticket Kiosk System

For the Ticket Kiosk System, the first UX target in Table 10-3 contains an objective UX measure for "Initial user performance." An obvious choice for the corresponding measuring instrument is a benchmark task. Here we need a simple and frequently used task that can be done in a short time by a casual new user in a walk-up ease-of-use situation. An appropriate benchmark task would involve buying tickets to an event. Here is a possible description to give the user participant:

> "BT1: Go to the Ticket Kiosk System and buy three tickets for the Monster Truck Pull on February 28 at 7:00 PM. Get three seats together as close to the front as possible. Pay with a major credit card."

In Table 10-4 we add this to the table as the measuring instrument for the first UX target.

Let us say we want to add another UX target for the "initial performance" UX measure, but this time we want to add some variety and use a different benchmark task as the measuring instrument—namely, the task of buying a movie ticket. In Table 10-5 we have entered this benchmark task in the second UX target, pushing the "first impression" UX target down by one.

Table 10-4

Choosing "buy special event ticket" benchmark task as measuring instrument for "initial performance" UX measure in first UX target

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT1: Buy special event ticket				
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression					

Table 10-5

Choosing "buy movie ticket" benchmark task as measuring instrument for second initial performance UX measure

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT1: Buy special event ticket				
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT2: Buy movie ticket				
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression					

How many benchmark tasks and UX targets do you need?

As in most things regarding human–computer interaction, it depends. The size and complexity of the system should be reflected in the quantity and complexity of the benchmark tasks and UX targets. We cannot even give you an estimate of a typical number of benchmark tasks.

You have to use your engineering judgment and make enough benchmark tasks for reasonable, representative coverage without overburdening the evaluation process. If you are new to this, we can say that we have often seen a dozen UX targets, but 50 would probably be too much—not worth the cost to pursue in evaluation.

How long should your benchmark tasks be (in terms of time to perform)? The typical benchmark task takes a range of a couple of minutes to 10 or 15 minutes. Some short and some long are good. Longer sequences of related tasks are needed to evaluate transitions among tasks. Try to avoid really long benchmark tasks because they may be tiring to participants and evaluators during testing.

Ensure ecological validity

The extent to which your evaluation setup matches the user's real work context is called *ecological validity* (Thomas & Kellogg, 1989). One of the valid criticisms of lab-based user experience testing is that a UX lab can be kind of a sterile environment, not a realistic setting for the user and the tasks. But you can take steps to add ecological validity by asking yourself, as you

write your benchmark task descriptions, how can the setting be made more realistic?

- What are constraints in user or work context?
- Does the task involve more than one person or role?
- Does the task require a telephone or other physical props?
- Does the task involve background noise?
- Does the task involve interference or interruption?
- Does the user have to deal with multiple simultaneous inputs, for example, multiple audio feeds through headsets?

As an example for a task that might be triggered by a telephone call, instead of writing your benchmark task description on a piece of paper, try calling the participant on a telephone with a request that will trigger the desired task. Rarely do task triggers arrive written on a piece of paper someone hands you. Of course, you will have to translate the usual boring imperative statements of the benchmark task description to a more lively and realistic dialogue: "Hi, I am Fred Ferbergen and I have an appointment with Dr. Strangeglove for a physical exam tomorrow, but I have to be out of town. Can you change my appointment to next week?"

Telephones can be used in other ways, too, to add realism to work context. A second telephone ringing incessantly at the desk next door or someone talking loudly on the phone next door can add realistic task distraction that you would not get from a "pure" lab-based evaluation.

Example: Ecological Validity in Benchmark Tasks for the Ticket Kiosk System

To evaluate use of the Ticket Kiosk System to manage the work activity of ticket buying, you can make good use of physical prototypes and representative locations. By this we mean building a touchscreen display into a cardboard or wooden kiosk structure and place it in the hallway of a relatively busy work area. Users will be subject to the gawking and questions of curiosity seekers. Having co-workers join the kiosk queue will add extra realism.

10.6.2 User Satisfaction Questionnaires

As a measuring instrument for a subjective UX measure, a questionnaire related to various user interaction design features can be used to determine a user's satisfaction with the interaction design. Measuring a user's satisfaction provides a subjective, but still quantitative, UX metric for the related UX measure.

As an aside, we should point out that objective and subjective measures are not always orthogonal.

As an example of a way they can intertwine, user satisfaction can actually affect user performance over a long period of time. The better users like the system, the more likely they are to experience good performance with it over the long term. In the following examples we use the QUIS questionnaire (description in Chapter 12), but there are other excellent choices, including the System Usability Scale or SUS (description in Chapter 12).

Example: Questionnaire as Measuring Instrument for the Ticket Kiosk System

If you think the first two benchmark tasks (buying tickets) make a good foundation for assessing the "first-impression" UX measure, then you can specify that a particular user satisfaction questionnaire or a specific subset thereof be administered following those two initial tasks, stipulating it as the measuring instrument in the third UX target of the growing UX target table, as we have done in Table 10-6.

Example: Goals, Measures, and Measuring Instruments

Before moving on to UX metrics, in Table 10-7 we show some examples of the close connections among UX goals, UX measures, and measuring instruments.

Table 10-6

Choosing questionnaire as measuring instrument for first-impression UX measure

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT1: Buy special event ticket				
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT2: Buy movie ticket				
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression	Questions Q1–Q10 in the QUIS questionnaire				

Table 10-7

Close connections among UX goals, UX measures, and measuring instruments

UX Goal	UX Measure	Potential Metrics
Ease of first-time use	Initial performance	Time on task
Ease of learning	Learnability	Time on task or error rate, after given amount of use and compared with initial performance
High performance for experienced users	Long-term performance	Time and error rates
Low error rates	Error-related performance	Error rates
Error avoidance in safety critical tasks	Task-specific error performance	Error count, with strict target levels (much more important than time on task)
Error recovery performance	Task-specific time performance	Time on recovery portion of the task
Overall user satisfaction	User satisfaction	Average score on questionnaire
User attraction to product	User opinion of attractiveness	Average score on questionnaire, with questions focused on the effectiveness of the "draw" factor
Quality of user experience	User opinion of overall experience	Average score on questionnaire, with questions focused on quality of the overall user experience, including specific points about your product that might be associated most closely with emotional impact factors
Overall user satisfaction	User satisfaction	Average score on questionnaire, with questions focusing on willingness to be a repeat customer and to recommend product to others
Continuing ability of users to perform without relearning	Retainability	Time on task and error rates re-evaluated after a period of time off (e.g., a week)
Avoid having user walk away in dissatisfaction	User satisfaction, especially initial satisfaction	Average score on questionnaire, with questions focusing on initial impressions and satisfaction

10.7 UX METRICS

A *UX metric* describes the kind of value to be obtained for a UX measure. It states what is being measured. There can be more than one metric for a given measure. As an example from the software engineering world, software complexity is a

measure; one metric for the software complexity measure (one way to obtain values for the measure) is "counting lines of code."

Most commonly, UX metrics are objective, performance-oriented, and taken while the participant is doing a benchmark task. Other UX metrics can be subjective, based on a rating or score computed from questionnaire results. Typical objective UX metrics include time to complete task[1] and number of errors made by the user. Others include frequency of help or documentation use; time spent in errors and recovery; number of repetitions of failed commands (what are users trying to tell us by repeating an action that did not work before?); and the number of commands, mouse-clicks, or other user actions to perform task(s).

If you are feeling adventurous you can use a count of the number of times the user expresses frustration or satisfaction (the "aha and cuss count") during his or her first session as an indicator of his or her initial impression of the interaction design. Of course, because the number of remarks is directly related to the length of the session, plan your levels accordingly or you can set your levels as a count per unit time, such as comments per minute, to factor out the time differences. Admittedly, this measuring instrument is rather participant dependent, depending on how demonstrative a participant feels during a session, whether a participant is generally a complainer, and so on, but this metric can produce some interesting results.

Typically, subjective UX metrics will represent the kind of numeric outcome you want from a questionnaire, usually based on simple arithmetic statistical measures such as the numeric average. Remember that you are going only for an engineering indicator of user experience, not for statistical significance.

Interestingly, user perceptions of elapsed time, captured via a questionnaire or post-session interview, can sometimes be an important UX measure. We know of such a case that occurred during evaluation of a new software installation procedure. The old installation procedure required the user to perform repeated disk (CD-ROM) swaps during installation, while the new installation procedure required only one swap. Although the new procedure took less time, users *thought* it took them longer because they were not kept busy swapping disks.

And do not overlook a combination of measures for situations where you have performance trade-offs. If you specify your UX metric as some function, such as a sum or an average, of two other performance-related metrics, for

[1]Although the time on task often makes a useful UX metric, it clearly is not appropriate in some cases. For example, if the task performance time is affected by factors beyond the user's control, then time on task is not a good measure of user performance. This exception includes cases of long and/or unpredictable communication and response-time delays, such as might be experienced in some Website usage.

example, time on task and error rate, you are saying that you are willing to give up some performance in one area if you get more in the other.

We hope you will explore many other possibilities for UX metrics, extending beyond what we have mentioned here, including:

- percentage of task completed in a given time
- ratio of successes to failures
- time spent moving cursor (would have to be measured using software instrumentation, but would give information about the efficiency of such physical actions, necessary for some specialized applications)
- for visibility and other issues, fixations on the screen, cognitive load as indicated by correlation to pupil diameter, and so on using eye-tracking

Finally, be sure you match up your UX measures, measuring instruments, and metrics to make sense in a UX target. For example, if you plan to use a questionnaire in a UX target, do not call the UX measure "initial performance." A questionnaire does not measure performance; it measures user satisfaction or opinion.

Example: UX Metrics for the Ticket Kiosk System

For the initial performance UX measure in the first UX target of Table 10-6, as already discussed in the previous section, the length of time to buy a special event ticket is an appropriate value to measure. We specify this by adding "time on task" as the metric in the first UX target of Table 10-8.

Table 10-8

Choosing UX metrics for UX measures

Work Role: User Class	UX goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT1: Buy special event ticket	Average time on task			
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT2: Buy movie ticket	Average number of errors			
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression	Questions Q1–Q10 in the QUIS questionnaire	Average rating across users and across questions			

CHAPTER

Prototyping 11

Objectives

After reading this chapter, you will:

1. Be able to articulate what prototyping is and why it is needed
2. Understand how to choose the appropriate depth and breadth, level of fidelity, and amount of interactivity of prototypes
3. Understand special types of prototypes, such as physical mockups and Wizard of Oz prototypes
4. Understand the appropriate type of prototype for a given stage of design evolution
5. Understand the role of prototypes in the transition to a product
6. Know how to make effective paper prototypes

11.1 INTRODUCTION

11.1.1 You Are Here

We begin each process chapter with a "you are here" picture of the chapter topic in the context of the overall Wheel lifecycle template; see Figure 11-1. Although prototyping is a kind of implementation, design and prototyping in practice often overlap and occur simultaneously. A prototype in that sense is a design representation.

So, as you create the design and its representation, you are creating the prototype. Therefore, although in Figure 11-1 it might seem that prototyping is limited to a particular place within a cycle of other process activities, like all other activities, prototyping does not happen only at some point in a rigid sequence.

11.1.2 A Dilemma, and a Solution

Have you ever rushed to deliver a product version without enough time to check it out? Then realized the design needed fixing? Sorry, but that ship has already left the station. The sooner you fail and understand why, the sooner you can succeed. As Frishberg (2006) tells us, "the faster you go, the sooner you know." If only you had made some early prototypes to work out the design changes before

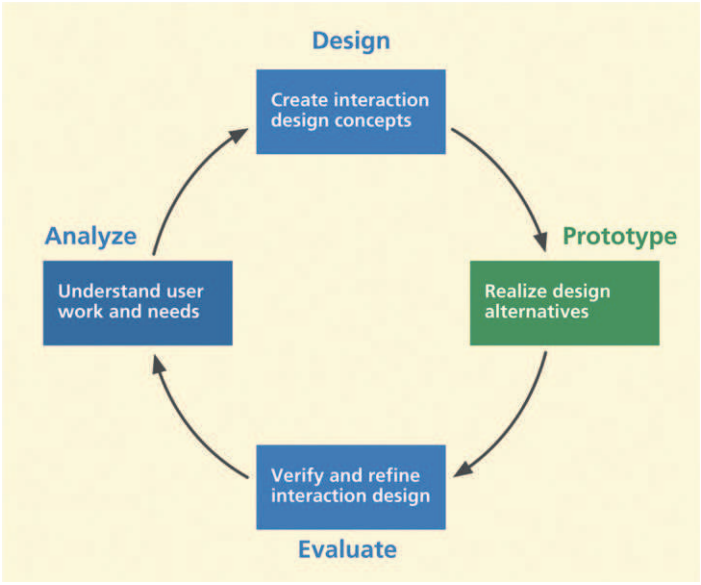


Figure 11-1

You are here; the chapter on prototyping in the context of the overall Wheel lifecycle template.

releasing it! In this chapter we show you how to use prototyping as a hatching oven for partially baked designs within the overall UX lifecycle process.

Traditional development approaches such as the waterfall method were heavyweight processes that required enormous investment of time, money, and personnel. Those linear development processes have tended to force a commitment to significant amounts of design detail without any means for visualizing and evaluating the product until it was too late to make any major changes.

Construction and modification of software by ordinary programming techniques in the past have been notoriously expensive and time-consuming activities. Little wonder there have been so many failed software development projects (Cobb, 1995; The Standish Group, 1994, 2001)—wrong requirements, not meeting requirements, imbalanced emphasis within functionality, poor user experience, and so much customer and user dissatisfaction.

In thinking about how to overcome these problems, we are faced with a dilemma. The only way to be sure that your system design is the right design and that your design is the best it can be is to evaluate it with real users. However, at the beginning you have a design but no system yet to evaluate. But after it is implemented, changes are much more difficult.

Enter the prototype. A prototype gives you something to evaluate before you have to commit resources to build the real thing. Because prototyping provides an early version of the system that can be constructed much faster and is less expensive, something to stand in stead of the real system to evaluate and inform refinement of the design, it has become a principal technique of the iterative lifecycle.

Universality of prototyping

The idea of prototyping is timeless and universal. Automobile designers build and test mockups, architects and sculptors make models, circuit designers use "bread-boards," artists work with sketches, and aircraft designers build and fly

experimental designs. Even Leonardo da Vinci and Alexander Graham Bell made prototypes.

Thomas Edison sometimes made 10,000 prototypes before getting just the right design. In each case the concept of a prototype was the key to affording the design team and others an early ability to observe something about the final product—evaluating ideas, weighing alternatives, and seeing what works and what does not.

Alfred Hitchcock, master of dramatic dialogue design, is known for using prototyping to refine the plots of his movies. Hitchcock would tell variations of stories at cocktail parties and observe reactions of his listeners. He would experiment with various sequences and mechanisms for revealing the story line. Refinement of the story was based on listener reactions as an evaluation criterion. *Psycho* is a notable example of the results of this technique.

Scandinavian origins

Like a large number of other parts of this overall lifecycle process, the origins of prototyping, especially low-fidelity prototyping, go back to the Scandinavian work activity theory research and practice of Ehn, Kyng, and others (Bjerknes, Ehn, & Kyng, 1987; Ehn, 1988) and participatory design work (Kyng, 1994). These formative works emphasized the need to foster early and detailed communication about design and participation in understanding the requirements for that design.

11.2 DEPTH AND BREADTH OF A PROTOTYPE

The idea of prototypes is to provide a fast and easily changed early view of the envisioned interaction design. To be fast and easily changed, a prototype must be something less than the real system. The choices for your approach to prototyping are about *how* to make it less. You can make it less by focusing on just the breadth or just the depth of the system or by focusing on less than full fidelity of details in the prototype (discussed later in this chapter).

11.2.1 Horizontal vs. Vertical Prototypes

Horizontal and vertical prototypes represent the difference between slicing the system by breadth and by depth in the features and functionality of a prototype (Hartson & Smith, 1991). Nielsen (1987) also describes types of prototypes based on how a target system is sliced in the prototype. In his usability

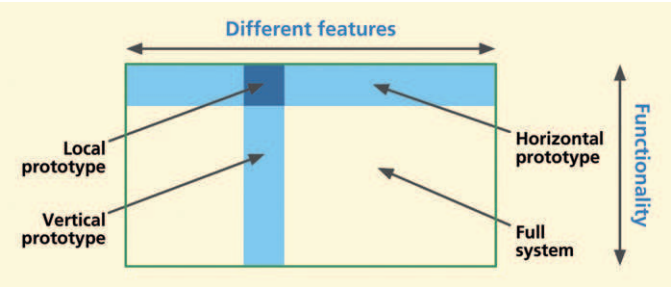


Figure 11-2
Horizontal and vertical prototyping concepts, from Nielsen (1993), with permission.

engineering book (1993), Nielsen illustrates the relative concepts of horizontal and vertical prototyping, which we show as Figure 11-2.

A horizontal prototype is very broad in the features it incorporates, but offers less depth in its coverage of functionality. A vertical prototype contains as much depth of functionality as possible in the current state of progress, but only for a narrow breadth of features.

A horizontal prototype is a good place to start with your prototyping, as it provides an overview on which you can base a top-down approach. A horizontal prototype is effective in demonstrating the product concept and for conveying an early product overview to managers, customers, and users (Kensing & Munk-Madsen, 1993) but, because of the lack of details in depth, horizontal prototypes usually do not support complete workflows, and user experience evaluation with this kind of prototype is generally less realistic.

A horizontal prototype can also be used to explore how much functionality will really be used by a certain class of users to expose typical users to the breadth of proposed functionality and get feedback on which functions would be used or not.

A vertical prototype allows testing a limited range of features but those functions that are included are evolved in enough detail to support realistic user experience evaluation. Often the functionality of a vertical prototype can include a stub for or an actual working back-end database.

A vertical prototype is ideal for times when you need to represent completely the details of an isolated part of an individual interaction workflow in order to understand how those details play out in actual usage. For example, you may wish to study a new design for the checkout part of the workflow for an e-commerce Website. A vertical prototype would show that one task sequence and associated user actions, in depth.

11.2.2 "T" Prototypes

A "T" prototype combines the advantages of both horizontal and vertical, offering a good compromise for system evaluation. Much of the interface is realized at a shallow level (the horizontal top of the T), but a few parts are done in depth (the vertical part of the T). This makes a T prototype essentially a

horizontal prototype, but with the functionality details filled out vertically for some parts of the design.

In the early going, the T prototype provides a nice balance between the two extremes, giving you some advantages of each. Once you have established a system overview in your horizontal prototype, as a practical matter the T prototype is the next step toward achieving some depth. In time, the horizontal foundation supports evolving vertical growth across the whole prototype.

11.2.3 Local Prototypes

We call the small area where horizontal and vertical slices intersect a "local prototype" because the depth and breadth are both limited to a very localized interaction design issue. A local prototype is used to evaluate design alternatives for particular isolated interaction details, such as the appearance of an icon, wording of a message, or behavior of an individual function. It is so narrow and shallow that it is about just one isolated design issue and it does not support any depth of task flow.

A local prototype is the solution for those times when your design team encounters an impasse in design discussions where, after a while, there is no agreement and people are starting to repeat themselves. Contextual data are not clear on the question and further arguing is a waste of time. It is time to put the specific design issue on a list for testing, letting the user or customer speak to it in a kind of "feature face-off" to help decide among the alternatives.

For example, your design team might not be able to agree on the details of a "Save" dialogue box and you want to compare two different approaches. So you can mockup the two dialogue box designs and ask for user opinions about how they behave.

Local prototypes are used independently from other prototypes and have very short life spans, useful only briefly when specific details of one or two particular design issues are being worked out. If a bit more depth or breadth becomes needed in the process, a local prototype can easily grow into a horizontal, vertical, or T prototype.

11.3 FIDELITY OF PROTOTYPES

The level of fidelity of a prototype is another dimension along which prototype content can be controlled. The fidelity of a prototype reflects how "finished" it is perceived to be by customers and users, not how authentic or correct the underlying code is (Tullis, 1990).

11.3.1 Low-Fidelity Prototypes

Low-fidelity prototypes are, as the term implies, prototypes that are not faithful representations of the details of look, feel, and behavior, but give rather high-level, more abstract impressions of the intended design. Low-fidelity prototypes are appropriate when design details have not been decided or when they are likely to change and it is a waste of effort and maybe even misleading to try and flesh out the details.

Because low-fidelity prototypes are sometimes not taken seriously, the case for low-fidelity prototyping, especially using paper, bears some explaining. In fact, it is perhaps at this lowest end of the fidelity spectrum, paper prototypes, that dwells the highest potential ratio of value in user experience gained per unit of effort expended. A low-fidelity prototype is much less evolved and therefore far less expensive. It can be constructed and iterated in a fraction of the time it takes to produce a good high-fidelity prototype.

But can a low-fidelity prototype, a prototype that does not look like the final system, really work? The experience of many has shown that despite the vast difference between a prototype and the finished product, low-fidelity prototypes can be surprisingly effective.

Virzi, Sokolov, and Karis (1996) found that people, customers, and users do take paper prototypes seriously and that low-fidelity prototypes do reveal many user experience problems, including the more severe problems. You can get your project team to take them seriously, too. Your team may be reluctant about doing a "kindergarten" activity, but they will see that users and customers love them and that they have discovered a powerful tool for their design projects.

But will not the low-fidelity appearance bias users about the perceived user experience? Apparently not, according to Wiklund, Thurrott, and Dumas (1992), who concluded in a study that aesthetic quality (level of finish) did not bias users (positively or negatively) about the prototype's perceived user experience. As long as they understand what you are doing and why, they will go along with it.

Sometimes it takes a successful personal experience to overcome a bias against low fidelity. In one of our UX classes, we had an experienced software developer who did not believe in using low-fidelity prototypes. Because it was a requirement in the project for the course, he did use the technique anyway, and it was an eye-opener for him, as this email he sent us a few months later attests:

> After doing some of the tests I have to concede that paper prototypes are useful. Reviewing screenshots with the customer did not catch some pretty obvious usability problems and now it is hard to modify the computer prototype. Another

> problem is that we did not get as complete a coverage with the screenshots of the system as we thought and had to improvise some functionality pretty quickly. I think someone had told me about that . . . .

Low-fidelity prototyping has long been a well-known design technique and, as Rettig (1994) says, if your organization or project team has not been using low-fidelity prototypes, you are in for a pleasant surprise; it can be a big breakthrough tool for you.

11.3.2 Medium-Fidelity Prototypes

Sometimes you need a prototype with a level in between low fidelity and high fidelity. Sometimes you have to choose one level of fidelity to stick with because you do not have time or other resources for your prototype to evolve from low fidelity to high-fidelity. For teams that want a bit more fidelity in their design representations than you can get with paper and want to step up to computer-based representations, medium-fidelity prototypes can be the answer.

In Chapter 9, for example, this occurs about when you undertake intermediate design and early detailed design. As a mechanism for medium-fidelity prototypes, wireframes (also in Chapter 9) are an effective way to show layout and the breadth of user interface objects and are fast becoming the most popular approach in many development organizations.

11.3.3 High-Fidelity Prototypes

In contrast, high-fidelity prototypes are more detailed representations of designs, including details of appearance and interaction behavior. High-fidelity is required to evaluate design details and it is how the users can see the complete (in the sense of realism) design. High-fidelity prototypes are the vehicle for refining the design details to get them just right as they go into the final implementation.

As the term implies, a high-fidelity prototype is faithful to the details, the look, feel, and behavior of an interaction design and possibly even system functionality. A high-fidelity prototype, if and when you can afford the added expense and time to produce it, is still less expensive and faster than programming the final product and will be so much more realistic, more interactive, more responsive, and so much more representative of a real software product than a low-fidelity prototype. High-fidelity prototypes can also be useful as advance sales demos for marketing and even as demos for raising venture capital for the company.

An extreme case of a high-fidelity prototype is the fully-programmed, whole-system prototype, discussed soon later, including both interaction design and non-user-interface functionality working together. Whole system prototypes can be as expensive and time-consuming as an implementation of an early version of the system itself and entail a lot of the software engineering management issues of non-prototype system development, including UX and SE collaboration about system functionality and overall design much earlier in the project than usual.

11.4 INTERACTIVITY OF PROTOTYPES

The amount of interactivity allowed by a prototype is not independent of the level of fidelity. In general, high interactivity requires high-fidelity. Here we discuss various ways to accomplish interactivity within a prototype.

11.4.1 Scripted and "Click-Through" Prototypes

The first prototypes to have any "behavior," or ability to respond to user actions, are usually scripted prototypes, meaning programmed with a scripting language. Scripting languages are easy to learn and use and, being high-level languages, can be used to produce some kinds of behavior very rapidly. But they are not effective tools for implementing much functionality. So scripted prototypes will be low or medium fidelity, but they can produce nice live-action storyboards of screens.

A "click-through" prototype is a medium-fidelity prototype with some active links or buttons that allow sequencing through screens by clicking, but usually with no more functionality than that. Wireframes can be used to make click-through prototypes by adding links that respond in simple ways to clicking, such as moving to the next screen.

11.4.2 A Fully Programmed Prototype

Even the prototypes of large systems can themselves be large and complex. On rare occasions and in very special circumstances, where time and resources permit and there is a genuine need, a project team is required to produce a high-fidelity full-system operational prototype of a large system, including at least some back-end functionality.

One such occasion did occur in the early 1990s when the FAA sought proposals from large development organizations for a big 10-year air traffic control system development project. Bidders successful in the first phase of

proposals would be required to design and build a full-function proof-of-concept prototype in a project that itself took nearly 2 years and cost millions of dollars. On the basis of this prototype phase, even larger multiyear contracts would be awarded for construction of the real system.

Such large and fully functional prototypes call for the power of a real programming language. Although the resulting prototype is still not intended to be the final system, a real programming language gives the most flexibility to produce exactly the desired look and feel. And, of course, a real programming language is essential for implementing extensive functionality. The process, of course, will not be as fast, low in cost, or easy to change.

11.4.3 "Wizard of Oz" Prototypes: Pay No Attention to the Man Behind the Curtain

The Wizard of Oz prototyping technique is a deceptively simple approach to the appearance of a high degree of interactivity and highly flexible prototype behavior in complex situations where user inputs are unpredictable. The setup requires two connected computers, each in a different room. The user's computer is connected as a "slave" to the evaluator's computer. The user makes input actions on one computer, which are sent directly to a human team member at the evaluator's computer, hidden in the second room.

The human evaluator sees the user inputs on the hidden computer and sends appropriate simulated output back to the user's computer. This approach has particular advantages, one of which is the apparently high level of interactivity as seen by the user. It is especially effective when flexible and adaptive "computer" behavior is of the essence, as with artificial intelligence and other difficult-to-implement systems. Within the limits of the cleverness of the human evaluator, the "system" should never break down or crash.

In one of the earliest uses of the Wizard of Oz technique that we know of, Good and colleagues (1984) designed empirically a command-driven email interface to accommodate natural novice user actions. Users were given no menus, help, no documentation, and no instruction.

Users were unaware that a hidden operator was intercepting commands when the system itself could not interpret the input. The design was modified iteratively so that it would have recognized and responded to previously intercepted inputs. The design progressed from recognizing only 7% of inputs to recognizing about 76% of user commands.

The Wizard of Oz prototyping technique is especially useful when your design ideas are still wide open and you want to see how users behave naturally in the course of simulated interaction. It could work well, for example, with a kiosk.

You would set up the general scope of usage expected and let users at it. You will see what they want to do. Because you have a human at the other end, you do not have to worry about whether you programmed the application to handle any given situation.

11.4.4 Physical MockUps for Physical Interactivity

If a primary characteristic of a product or system is physicality, such as you have with a handheld device, then an effective prototype will also have to offer physical interactivity. Programming new applications on physical devices with real software means complex and lengthy implementation on a challenging hardware and software platform. Prototypes afford designers and others insight into the product look and feel without complicated specialized device programming.

Some products or devices are "physical" in the sense that they are something like a mobile device that users might hold in their hands. Or a system might be "physical" like a kiosk. A physical prototype for such products goes beyond screen simulation on a computer; the prototype encompasses the whole device. Pering (2002) describes a case study of such an approach for a handheld communicator device that combines the functionality of a PDA and a cellphone.

If the product is to be handheld, make a prototype from cardboard, wood, or metal that can also be handheld. If the product, such as a kiosk, is to sit on the floor, put the prototype in a cardboard box and add physical buttons or a touchscreen.

You can use materials at hand or craft the prototype with realistic hardware. Start off with glued-on shirt buttons and progress to real push-button switches. Scrounge up hardware buttons and other controls that are as close to those in your envisioned design as possible: push buttons, tilt buttons, sliders, for example, from a light dimmer, knobs and dials, rocker switch, or a joystick from an old Nintendo game.

Even if details are low fidelity, these are higher fidelity in some ways because they are typically 3D, embodied, and tangible. You can hold them in your hands. You can touch them and manipulate them physically. Also, physical prototypes are excellent media for supporting evaluation of emotional impact and other user experience characteristics beyond just usability.

And just because physical product prototyping usually involves a model of physical hardware does not rule out being a low-fidelity prototype. Designers of the original Palm PDA carried around a block of wood as a physical prototype of the envisioned personal digital assistant. They used it to explore the physical

feel and other requirements for such a device and its interaction possibilities (Moggridge, 2007, p. 204).

Physical prototyping is now being used for cellphones, PDAs, consumer electronics, and products beyond interactive electronics, employing found objects, "junk" (paper plates, pipe cleaners, and other playful materials) from the recycle bin, thrift stores, dollar stores, and school supply shops (N. Frishberg, 2006). Perhaps IDEO[1] is the company most famous for its physical prototyping for product ideation; see their shopping cart project video (*ABC News Nightline*, 1999) for a good example.

Wright (2005) describes the power of a physical mockup that users can see and hold as a real object over just pictures on a screen, however powerful and fancy the graphics. Users get a real feeling that this *is* the product. The kind of embodied user experience projected by this approach can lead to a product that generates user surprise and delight, product praise in the media, and must-have cachet in the market.

Paper-in-device mockup prototype, especially for mobile applications

The usual paper prototype needs an "executor," a person playing computer to change screens and do all the other actions of the system in response to a user's actions. This role of mediator between user and device will necessarily interfere with the usage experience, especially when a large part of that experience involves, holding, feeling, and manipulating the device itself.

Bolchini, Pulido, and Faiola (2009) and others devised a solution by which they placed the paper prototype inside the device, leveraging the advantages of paper prototyping in evaluating mobile device interfaces with the real physical device. They drew the prototype screens on paper, scanned them, and loaded them into the device as a sequence of digital images that the device can display. During evaluation, users can move through this sequential navigation by making touches or gestures that the device already can recognize.

This is an agile and inexpensive technique, and the authors reported that their testing showed that even this limited amount of interactivity generated a lot of useful feedback and discussion with evaluation users. Also, by adding page annotations about user interactions, possible user thoughts, and other behind-the-scenes information, the progression of pages can become like a design storyboard of the usage scenario.

[1] http://www.ideo.com

11.4.5 Animated Prototypes

Most prototypes are static in that they depend on user interaction to show what they can do. Video animation can bring a prototype to life for concept demos, to visualize new interaction designs, and to communicate design ideas. While animated prototypes are not interactive, they are at least active.

Löwgren (2004) shows how video animations based on a series of sketches can carry the advantages of low-fidelity prototypes to new dimensions where a static paper prototype cannot tread. Animated sketches are still "rough" enough to invite engagement and design suggestions but, being more like scenarios or storyboards, animations can convey flow and sequencing better in the context of usage.

HCI designers have been using video to bring prototypes to life as early as the 1980s (Vertelney, 1989). A simple approach is to use storyboard frames in a "flip book" style sequence on video or, if you already have a fairly complete low-fidelity prototype, you can film it in motion by making a kind of "claymation" frame-by-frame video of its parts moving within an interaction task.

11.5 CHOOSING THE RIGHT BREADTH, DEPTH, LEVEL OF FIDELITY, AND AMOUNT OF INTERACTIVITY

There are two major factors to consider when choosing the right breadth, depth, level of fidelity, and amount of interactivity of your prototypes: the stage of progress you are in within the overall project and the design perspective in which you are prototyping. These two factors are interrelated, as stages of progress occur within each of the design perspectives.

11.5.1 Using the Right Level of Fidelity for the Current Stage of Progress

Choosing your audience and explaining the prototype

In general, low-fidelity prototypes are a tool to be used within the project team. Low-fidelity prototypes are shown to people outside the team only to get feedback on very specific aspects of the design. If low-fidelity prototypes are shown casually around to users and customer personnel without careful explanation, they can be misinterpreted. To someone not familiar with their use, a paper prototype can look like the product of an inexperienced amateur.

Even if they do get beyond the rough appearance, without guidance as to what kind of feedback you want, "sophisticated" users and customers will immediately see missing features and think that you do not know what you are doing, possibly creating a credibility gap. Therefore, low-fidelity prototypes are often considered "private" to the project team and reserved for your own use for early exploration and iterative refinement of the conceptual design and early workflow.

Therefore, when a project is deliverable-oriented and the customer expects to evaluate your progress based on what they see developing as a product, a medium- or high-fidelity prototype can be used as a design demo. Practitioners often construct pixel-perfect representations of envisioned designs for these prototypes to show off designs to customers, users, and other non-team stakeholders. Such realistic-looking demos, however, carry the risk of being interpreted as complete designs, as versions of the final product. If something is wrong or missing, the designers are still blamed. Explaining everything in advance can head off these complications.

A progression of increasing fidelity to match your stage of progress

As a general rule, as you move through stages of progress in your project, you will require increasing levels of fidelity in your prototypes. For example, the goal in an early stage might be to determine if your design approach is even a good idea or a feasible one.

The goal of a later stage might simply be to show off: "Look at what a cool design we have!" In Table 11-1 we describe the appropriate time and place to use each kind of prototype in terms of various kinds of iteration within design production (Chapter 9). The types of prototypes mentioned in Table 11-1 are described in various places, mostly in this chapter.

11.5.2 Using the Right Level of Fidelity for the Design Perspective Being Addressed

For each design perspective in which you make a prototype, you must decide which kind of prototype, horizontal or vertical and at what fidelity, is needed, requiring you to consider what aspects of the design you are worried about and what aspects need to be tested in that perspective. In large part, this means asking about the audience for and the purpose of your prototype in the context of that perspective. What do you hope to accomplish with a prototype in the design perspective being addressed? What questions will the prototype help you answer?

Table 11-1

Summary of the uses for various levels of fidelity and types of prototypes

Kind of Iteration	Purpose	Types of Prototypes
Ideation and sketching	To support exploring ideas, brainstorming, and discussion (so design details are inappropriate)	Sketches, fast and disposable mockups, ultralow fidelity
Conceptual design	To support exploration and creation of conceptual design, the high-level system structure, and the overall interaction metaphor	Evolution from hand-drawn paper, computer-printed paper, low-fidelity wireframes, high-fidelity wireframes, to pixel-perfect interactive mockups (to communicate with customer)
Intermediate design	To support interaction design for tasks and task threads	Evolution from paper to wireframes
Detailed design	Support for deciding navigation details, screen design and layout, including pixel-perfect visual comps complete specification for look and feel of the "skin"	Detailed wireframes and/or pixel-perfect interactive mockups
Design refinement	To support evaluation to refine a chosen design by finding and removing as many UX problems as possible	Medium to high fidelity, lots of design detail, possibly a programmed prototype

Ecological Perspective

The ecological design perspective is about how the system or product works within its external environment. It is about how the system or product is used in its context and how the system or product interacts or communicates with its environment in the process.

Prototyping for the ecological perspective

To support exploration of the high-level system structure, a prototype in the ecological perspective is a kind of concept map to how the different parts of the system will work at the conceptual level and how it fits in with the rest of the world—other systems and products and other users.

As you evaluate the conceptual design, remember that you are looking at the big picture so the prototypes do not need to be high fidelity or detailed. If evaluation with early conceptual prototypes shows that users do not get along well with the basic metaphor, then the designers will not have wasted all the time it takes to work out design details of interaction objects such as screen icons, messages, and so on.

The development of IBM's Olympic Message System (Gould et al., 1987) was an avant garde example of product prototyping with emphasis on the ecological setting. IBM was tasked to provide a communications system for the 1984 Olympics in Los Angeles to keep athletes, officials, families, and friends in immediate contact during the games. For their earliest concept testing they used a "Wizard of Oz" technique whereby participants pressed keys on a computer terminal and spoke outgoing messages. The experimenter read aloud the incoming messages and gave other outputs as the interaction required.

For enhanced ecological validity they used a "hallway methodology" that started with a hollow wooden cylinder set in IBM building hallways, with pictures of screens and controls pasted on. They quickly learned a lot about the best height, location, labeling, and wording for displays. Real interactive displays housed in more finished kiosk prototypes led to even more feedback from visitors and corporate passersby. The resulting system was a big success at the Olympics.

Prototyping for the interaction perspective

For conceptual design, support early exploration with ideation and sketching using rapid and disposable low-fidelity prototypes. As you evaluate the conceptual design, remember that you are looking at the big picture so the fidelity of prototypes can be low. Use many rapid iterations to refine candidate conceptual design ideas.

As you move into intermediate design iteration, start by choosing a few tasks that are the most important and prototype them fairly completely. Mockup a typical task so that a user can follow a representative task thread.

Use medium-fidelity prototypes, such as wireframes, to flesh out behavior, including sequencing and responses to user actions. As we will see in later chapters on formative evaluation, a great deal can be learned from an incomplete design in a prototype.

For detailed design, after you have exhausted possibilities in evaluating the conceptual model and early screen design ideas with your low-fidelity, possibly paper, prototype, you will move on. You might next use a computer-printed paper prototype or a computer-based mockup to test increasing amounts of design detail.

You will flesh out your prototype with more complete task threads, well-designed icons, and carefully worded messages. Representing and evaluating full design details require high-fidelity prototypes, possibly programmed and possibly connected with some working functionality, such as database functions.

Prototyping for the emotional perspective

A prototype to support evaluation of emotional impact needs certain kinds of details. High fidelity and high interactivity are usually required to support this perspective. Although full details at the interaction level may not always be required, you do need details relating to fun, joy of use, and user satisfaction. Further, the emotional perspective for physical devices more or less demands physical mockups for a real feeling of holding and manipulating the device.

Ecological Validity

Ecological validity refers to the realism with which a design of evaluation setup matches the user's real work context. It is about how accurately the design or evaluation reflects the relevant characteristics of the ecology of interaction, i.e., its context in the world or its environment.

Interaction Perspective

The interaction design perspective is about how users operate the system or product. It is a task and intention view, where user and system come together. It is where users look at displays and manipulate controls, doing sensory, cognitive, and physical actions.

Emotional Perspective

The emotional design perspective is about emotional impact and value-sensitive aspects of design. It is about social and cultural implications, as well as the aesthetics and joy of use.

11.5.3 Managing Risk and Cost within Stages of Progress and within Design Perspectives

There has been much debate over the relative merits of low-fidelity prototypes vs. high-fidelity prototypes, but Buxton (2007a) has put it in a better light: It is not so much about high-fidelity prototypes vs. low-fidelity prototypes as it is about getting the right prototype. But, of course, part of getting it right is in determining the right level of abstraction for the purpose.

One way to look at the horizontal vs. vertical and low-fidelity vs. high-fidelity question is with respect to the three design perspectives (Chapter 7). For each of these perspectives, it is about managing risk, particularly the risk (in terms of cost) of getting the design wrong (with respect to the corresponding perspective) and the cost of having to change it.

A user interaction design can be thought of in two parts:

- the appearance, especially the visual aspects of the user interface objects
- the behavior, including sequencing and responses to user actions

Of these, which has the biggest risk in terms of cost to change late in the schedule? It is the behavior and sequencing. The behavior is the part that corresponds roughly to the design metaphor envisioned to support the user workflow. Therefore, we should try to get the best design we can for the behavior before we worry about details of appearance. That means our earliest and easiest to change prototypes should represent interaction design behavior and that means having a low-fidelity prototype first. This interaction structure and sequencing is very easy to change with paper screens, but becomes increasingly more difficult to modify as it is committed to programming code.

In low-fidelity prototypes it can even be a disadvantage to show too many details that appear refined. As Rettig (1994) points out, if you have a nice slick look and feel, you will naturally get most of your feedback on the look and feel details rather than on high-level issues such as workflow, task flow, overall layout, and the metaphor. Also, some users may be less willing to suggest changes for a prototype that even appears to be high fidelity because of the impression that the design process is completed and that any feedback they provide is probably too late (Rudd, Stern, & Isensee, 1996).

Later, however, increasingly higher fidelity prototypes can be used to establish and refine the exact appearance, the visual and manipulation aspects of interface objects such as colors, fonts, button design, highlighting an object, and so on, and eventually to bring in some depth in terms of functionality, for example, more detail about checking and handling errors in user inputs. As shown in Table 11-2, there is a place for both low-fidelity and high-fidelity prototypes in most design projects.

Type of Prototype	"Strength"	When in Lifecycle to Apply "Strength"	Cost to Fix Appearance	Cost to Fix Sequencing
Low fidelity (e.g., paper)	Flexibility; easy to change sequencing, overall behavior	Early	Almost none	Low
High fidelity (e.g., computer)	Fidelity of appearance	Later	Intermediate	High

Table 11-2
Summary of comparison of low-fidelity and high-fidelity prototypes

Finally, and just as an aside, prototyping is a technique that can help manage and reduce risks on the software engineering side as well on the UX side of a project.

11.5.4 Summary of the Effects of Breadth, Depth, and Fidelity Factors

In the graph in Figure 11-3 we show roughly how scope (vertical vs. horizontal) and fidelity issues play out in the choice of prototyping approaches based on what the designer needs.

11.6 PAPER PROTOTYPES

Soon after you have a conceptual design mapped out, give it life as a low-fidelity prototype and try out the concept. This is the time to start with a horizontal prototype, showing the possible breadth of features without much depth. The facility of paper prototypes enables you, in a day or two, to create a new design idea, implement it in a prototype, evaluate it with users, and modify it.

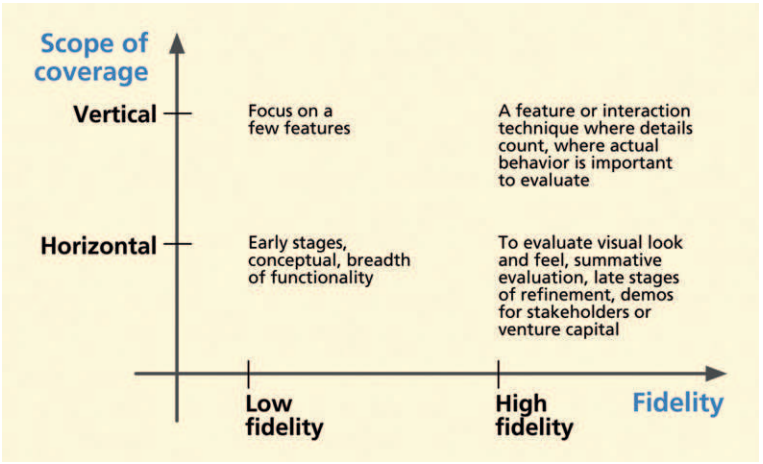


Figure 11-3
Depth, breadth, and fidelity considerations in choosing a type of prototype.

Low fidelity usually means paper prototypes. You should construct your early paper prototypes as quickly and efficiently as possible. Early versions are just about interaction, not functionality. You do not even have to use "real" widgets.

Sometimes a paper prototype can act as a "coding blocker" to prevent time wasted on coding too early. At this critical juncture, when the design is starting to come together,

programmers are likely to suffer from the WISCY syndrome (Why Isn't Sam Coding Yet?). They naturally want to run off and start coding.

You need a way to keep people from writing code until we can get the design to the point where we should invest in it. Once any code gets written, there will be ownership attached and it will get protected and will stay around longer than it should. Even though it is just a prototype, people will begin to resist making changes to "their baby"; they will be too invested in it. And other team members, knowing that it is getting harder to get changes through, will be less willing to suggest changes.

11.6.1 Paper Prototypes for Design Reviews and Demos

Your earliest paper prototypes will have no functionality or interaction, no ability to respond to any user actions. You can demonstrate some predefined sequences of screen sketches as storyboards or "movies" of envisioned interaction. For the earliest design reviews, you just want to show what it looks like and a little of the sequencing behavior. The goal is to see some of the interaction design very quickly—in the time frame of hours, not days or weeks.

11.6.2 Hand-Drawn Paper Prototypes

The next level of paper prototypes will support some simulated "interaction." As the user views screens and pretends to click buttons, a human "executor" plays computer and moves paper pieces in response to those mock user actions.

11.6.3 Computer-Printed Paper Prototypes

Paper prototypes, with user interface objects and text on paper printed via a computer, are essentially the same as hand-drawn paper prototypes, except slightly higher fidelity in appearance. You get fast, easy, and effective prototypes with added realism at very low cost. To make computer-printable screens for low-fidelity prototypes, you can use tools such as OmniGraffle (for the Mac) or Microsoft Visio.

Berger (2006) describes one successful case of using a software tool not intended for prototyping. When used as a prototyping tool, Excel provides grid alignment for objects and text, tabbed pages to contain a library of designs, a hypertext feature used for interactive links, and a built-in primitive database capability.

Cells can contain graphical images, which can also be copied and pasted, thus the concept of templates for dialogue box, buttons, and so on can be thought of as native to Excel. Berger claimed fast turnarounds, typically on a daily basis.

11.6.4 Is not paper just a stopgap medium?

Is not paper prototyping a technique necessary just because we do not yet have good enough software prototyping tools? Yes and no. There is always hope for a future software prototyping tool that can match the fluency and spontaneity afforded by the paper medium. That would be a welcome tool indeed and perhaps wireframing is heading in that direction but, given the current software technology for programming prototypes even for low-fidelity prototypes, there is no comparison with the ease and speed with which paper prototypes can be modified and refined, even if changes are needed on the fly in the midst of an evaluation session.

Therefore, at least for the foreseeable future, paper prototyping has to be considered as more than just a stopgap measure or a low-tech substitute for that as yet chimerical software tool; it is a legitimate technology on its own.

Paper prototyping is an embodied effort that involves the brain in the creative hand–eye feedback loop. When you use any kind of programming, your brain is diverted from the design to the programming. When you are writing or drawing on the paper with your hands and your eyes and moving sheets of paper around manually, you are thinking about design. When you are programming, you are thinking about the software tool.

Rettig (1994) says that with paper, "... interface designers spend 95% of their time thinking about the design and only 5% thinking about the mechanisms of the tool. Software-based tools, no matter how well executed, reverse this ratio."

11.6.5 Why Not Just Program a Low-Fidelity Prototype?

At "run-time" (or evaluation time), it is often useful to write on the paper pages, something you cannot do with a programmed prototype. Also, we have found that paper has much broader visual bandwidth, which is a boon when you want to look at and compare multiple screens at once. When it comes time to change the interaction sequencing in a design, it is done faster and visualized more easily by shuffling paper on a table.

Another subtle difference is that a paper prototype is always available for "execution," but a software prototype is only intermittently executable—only between sessions of programming to make changes. Between versions, there is a need for fast turnaround to the next version, but the slightest error in the code will disable the prototype completely. Being software, your prototype is susceptible to a single bug that can bring it crashing down and you may be caught in a position where you have to debug in front of your demo audience or users.

The result of early programmed prototypes is almost always slow prototyping, not useful for evaluating numerous different alternatives while the trail to interaction design evolution is still hot. Fewer iterations are possible, with more "dead time" in between where users and evaluators can lose interest and have correspondingly less opportunity to participate in the design process. Also, of course, as the prototype grows in size, more and more delay is incurred from programming and keeping it executable.

Because programmed prototypes are not always immediately available for evaluation and design discussion, sometimes the prototyping activity cannot keep up with the need for fast iteration. Berger (2006) relates an anecdote about a project in which the user interface software developer had the job of implementing design sketches and design changes in a Web page production tool. It took about 2 weeks to convert the designs to active Web pages for the prototype and in the interim the design had already changed again and the beautiful prototypes were useless.

11.6.6 How to Make an Effective Paper Prototype

Almost all you ever wanted to know about prototyping, you learned in Kindergarten.

Get out your paper and pencil, some duct tape, and WD-40. Decide who on your team can be trusted with sharp instruments, and we are off on another adventure. There are many possible approaches to building paper prototypes. The following are some general guidelines that have worked for us and that we have refined over many, many iterations.

Start by setting a realistic deadline. This is one kind of activity that can go on forever. Time management is an important part of any prototyping activity. There is no end to the features, polishing, and tweaking that can be added to a paper prototype. And watch out for design iteration occurring before you even get the first prototype finished. You can go around in circles before you get user inputs and it probably will not add much value to the design. Why polish a feature that might well change within the next day anyway?

Gather a set of paper prototyping materials. As you work with paper prototypes, you will gather a set of construction materials customized to your approach. Here is a starter list to get you going:

- Blank plastic transparency sheets, 8½ × 11; the very inexpensive write-on kind works fine; you do not need the expensive copier-type plastic
- An assortment of different colored, fine-pointed, erasable and permanent marking pens

- A supply of plain copier-type paper (or a pad of plain, unlined, white paper)
- Assorted pencils and pens
- Scissors
- "Scotch" tape (with dispensers)
- A bottle of Wite-out or other correction fluid
- Rulers or straight edges
- A yellow and/or pink highlighter
- "Sticky" (e.g., Post-it) note pads in a variety of sizes and colors

Keep these in a box so that you have them handy for the next time you need to make a paper prototype.

Work fast and do not color within the lines. If they told you in school to use straight lines and color only within the boxes, here is a chance to revolt, a chance to heal your psyche. Change your personality and live dangerously, breaking the bonds of grade school tyranny and dogmatism, something you can gloat about in the usual postprototype cocktail party.

Draw on everything you have worked on so far for the design. Use your conceptual design, design scenarios, ideation, personas, storyboards, and everything else you have created in working up to this exciting moment of putting it into the first real materialization of your design ideas.

Make an easel to register (align) your screen and user interface object sheets of paper and plastic. Use an "easel" to register each interaction sheet with the others. The simple foam-core board easels we make for our short courses are economical and serviceable. On a piece of foam-core board slightly larger than 8½ × 11, on at least two (of the four) adjacent sides add some small pieces of additional foam-core board as "stops," as seen in Figures 11-4 and 11-5, against which each interaction sheet can be pushed to ensure proper positioning. When the prototype is being "executed" during UX evaluation, the easel will usually be taped to the tabletop for stability.

Figure 11-4

Foam-core board paper prototype easel with "stops" to align the interaction sheets.

Make underlying paper foundation "screens." Start with simplest possible background for each screen in pencil or pen

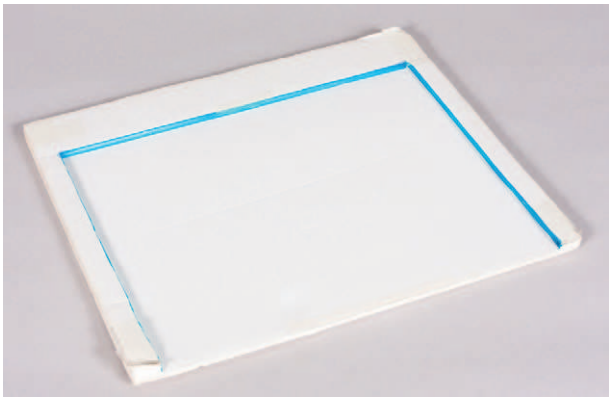


Figure 11-5
Another style of "stops" on a foam-core board paper prototype easel.

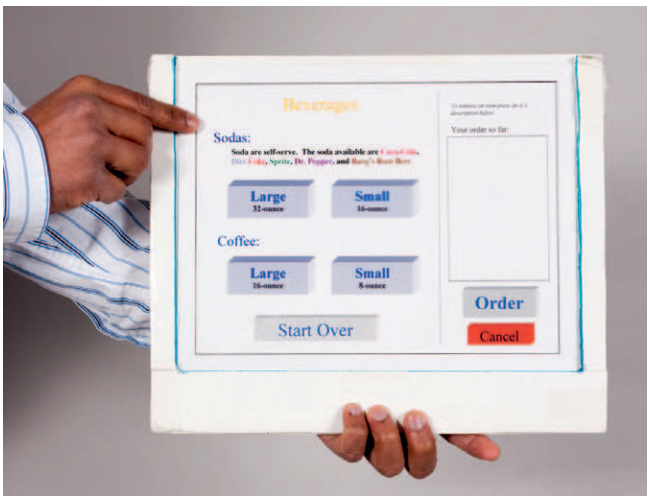
on full-size paper (usually 8½ × 11) as a base for all moving parts. Include only parts that never change. For example, in a calendar system prototype, show a monthly "grid," leaving a blank space for the month name). See Figure 11-6.

Use paper cutouts taped onto full-size plastic "interaction sheets" for all moving parts. Everything else, besides the paper foundation, will be taped to transparent plastic sheets. Draw everything else (e.g., interaction objects, highlights, dialogue boxes, labels) in pencil, pen, or colored markers on smaller pieces of paper and cut them out. Tape them onto separate full-size 8½ × 11 blank plastic sheets in the appropriate position aligned relative to objects in the foundation screen and to objects taped to other plastic sheets.

We call this full-size plastic sheet, with paper user interface object(s) taped in position, an "interaction sheet." The appearance of a given screen in your prototype is made up of multiple overlays of these interaction sheets. See Figure 11-7.

When these interaction sheets are aligned against the stops in the easel, they appear to be part of the user interface, as in the case of the pop-up dialogue box in Figure 11-8.

Figure 11-6
Underlying paper foundation "screen."



Be creative. Think broadly about how to add useful features to your prototype without too much extra effort. In addition to drawing by hand, you can use simple graphics or paint programs to import images such as buttons, and resize, label, and print them in color. Fasten some objects such as pull-down lists to the top or side of an interaction sheet with transparent tape hinges so that they can "flap down" to overlay the screen when they are selected. See Figure 11-9.

Scrolling can be done by cutting slits in your paper menu, which is taped to a plastic sheet. Then a slightly smaller slip

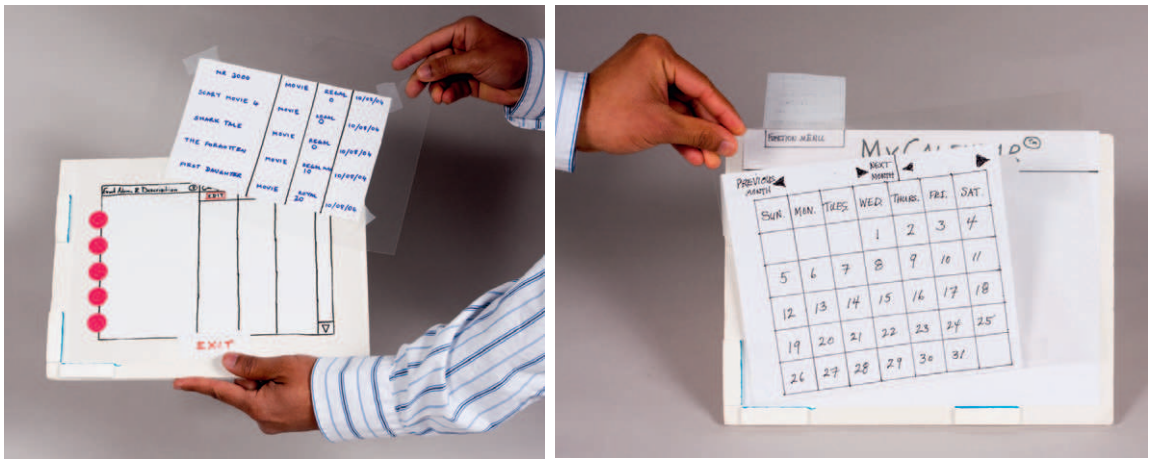


Figure 11-7

Paper cutouts taped to full-size plastic for moving parts.

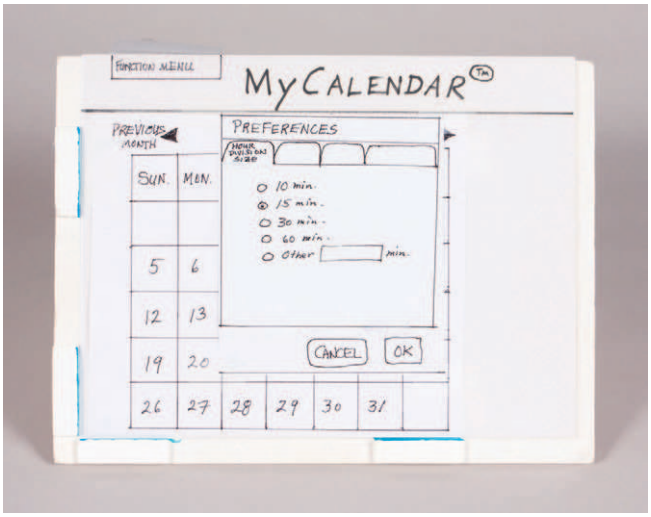
of paper with the menu choices can be slid through the slots. See Figure 11-10.

Use any creative techniques to demonstrate motion, dynamics, and feedback.

Do not write or mark on plastic interaction sheets. The plastic interaction sheets are almost exclusively for mounting and positioning the paper pieces. The plastic is supposed to be transparent; that is how layering works. Do not write or draw on the plastic. The only exception is for making transparent objects such as highlights or as an input medium on which users write input values. Later we will discuss completely blank sheets for writing inputs.

Figure 11-8

A "Preferences" dialogue box taped to plastic and aligned in easel.



Make highlights on plastic with "handles" for holding during prototype execution. Make a highlight to fit each major selectable object. Cut out a plastic square or rectangle with a long handle and color in the highlight (usually just an outline so as not to obscure the object or text being highlighted) with a permanent marking pen. See Figure 11-11.

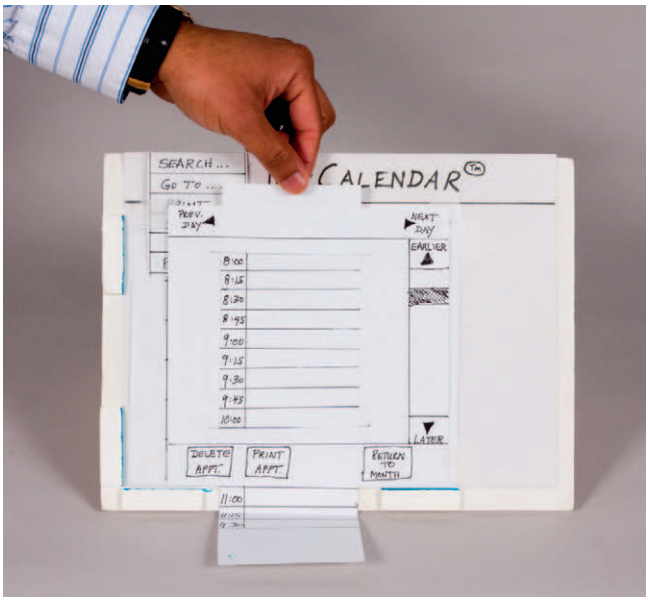
Figure 11-9

Pull-down menu on a tape "hinge."

Make your interaction sheets highly modular by including only a small amount on each one. Instead of customizing a single screen or page, build up each screen or display in layers. The less you put on each layer, the more modular and, therefore, the more reuse you will get. With every feature and every variation of appearance taped to a different sheet of plastic, you have the best chance at being able to show the most variation of appearances and user interface object configurations you might encounter. Be suspicious of a lot of writing/drawing on one interaction sheet. When a prototype user gives an input, it usually makes a change in the display. Each possible change should go on a separate interaction sheet.

Figure 11-10

Paper sliding through a slit for scrolling.



Get modularity by thinking about whatever needs to appear by itself. When you make an interaction sheet, ask yourself: *Will every single detail on here always appear together?* If there is a chance two items on the same interaction sheet will ever appear separately, it is best to put them on separate interaction sheets. They come back together when you overlay them together, but they can still be used separately, too. See Figure 11-12.

Do lots of sketching and storyboarding before making interaction sheets. This will save time and work.

Use every stratagem for minimizing work and time. Focus on design, not writing and paper cutting.

Reuse at every level. Make it a goal to not draw or write anything twice; use templates for the common parts of similar objects. Use a copy machine or scanner to reproduce common parts of similar interaction objects and write in only the differences. For example, for a calendar, use copies of a blank month template, filling in the days for each month. The idea is to capture in a template everything that does not have to change from one instance to another.

Figure 11-11

Selection highlight on plastic with a long handle.

Cut corners when it does not hurt things. Always trade off accuracy (when it is not needed) for efficiency (that is always needed). As an example, if it is not important to have the days and dates be exactly right for a given month on a calendar, use the same date numbers for each month in your early prototype. Then you can put the numbers in your month template and not have to write any in.

Figure 11-12

Lots of pieces of dialogue as paper cutouts aligned on plastic sheets.

Make the prototype support key tasks. Prototype at least all benchmark tasks from your UX target table, as this prototype will be used in the formative evaluation exercise.

Make a "this feature not yet implemented" message. This is the prototype's response to a user action that was not anticipated or that has not yet been included in the design. You will be surprised

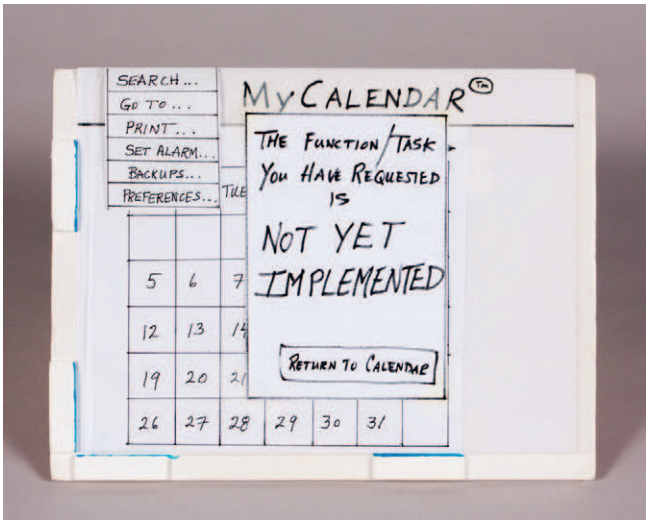


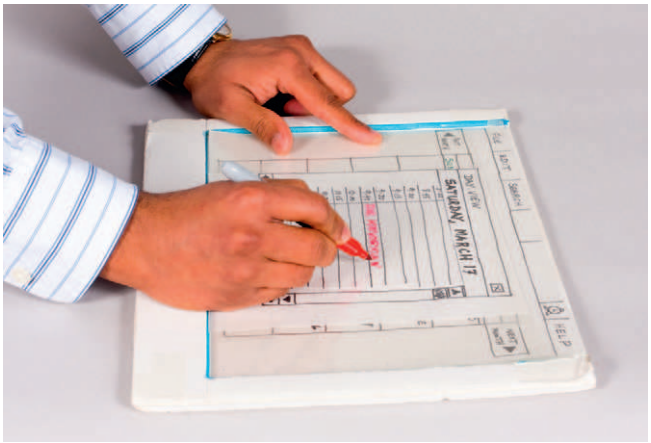
Figure 11-13
"Not yet implemented" message.

how often you may use this in user experience evaluation with early prototypes. See Figure 11-13.

Include "decoy" user interface objects. If you include only user interface objects needed to do your initial benchmark tasks, it may be unrealistically easy for users to do just those tasks. Doing user experience testing with this kind of initial interaction design does not give a good idea of the ease of use of the design when it is complete and contains many more user interface objects to choose from and many more other choices to make during a task.

Therefore, you should include many other "decoy" buttons, menu choices, etc., even if they do not do anything (so participants see more than just the "happy path" for their benchmark tasks). Your decoy objects should look plausible and should, as much as possible, anticipate other tasks and other paths. Users performing tasks with your prototype will be faced with a more realistic array of user interface objects about which they will have to think as they make choices about what user actions are next. And when they click on a decoy object, that is when you get to use your "not implemented" message. (Later, in the evaluation chapters, we while discuss probing the users on why they clicked on that object when it is not part of your envisioned task sequence.)

Figure 11-14
Data entry on clear plastic overlay sheet.



Accommodate data value entry by users. When users need to enter a value (e.g., a credit card number) into a paper prototype, it is usually sufficient to use a clear sheet of plastic (a blank interaction sheet) on top of the layers and let them write the value in with a marking pen; see Figure 11-14. Of course, if your design requires them to enter that number using a touchscreen on an iPad, for example, you have to create a "text input" interaction sheet.

Create a way to manage complex task threads. Before an evaluation session, the prototype "executor" will have all the paper sheets and overlays all lined up and ready to put on the easel in response to user actions. When the number of prototype pieces gets very large, however, it is difficult to know what stack of pieces to use at any point in the interaction, and it is even more difficult to clean it all up after the session to make it ready for the next session.

As an organizing technique that works most of the time, we have taken to attaching colored dots to the pieces, color coding them according to task threads. Sheets of adhesive-backed colored circles are available at most office supply stores. See Figure 11-15. Numbers written on the circles indicate the approximate expected order of usage in the corresponding task thread, which is the order to sort them in when cleaning up after a session.

Figure 11-15
Adhesive-backed circles for color coding task threads on prototype pieces.

Pilot test thoroughly. Before your prototype is ready to be used in real user experience evaluation sessions, you must give it a good shake-down. Pilot test your prototype to be sure that it will support all your benchmark tasks. You do not want to make the rookie mistake of "burning" a user participant (subject) by getting them started only to discover the prototype "blows up" and prevents benchmark task performance.

Simulate user experience evaluation conditions by having one member of your team "execute" the prototype while another member plays "user" and tries out all benchmark tasks. The user person should go through each task in as many ways as anyone thinks possible to head off the "oh, we never thought they would try *that*" syndrome later in testing. Do not assume error-free performance by your users; try to have appropriate error messages where user errors might occur. When you think your prototype is ready, get someone from outside your group and have them play the user role in more pilot testing.

Exercise

See Exercise 11-1, Building a Low-Fidelity Paper Prototype for Your System

CHAPTER

Mental Models and Conceptual Design

8

Objectives

After reading this chapter, you will:

1. Understand designers' and users' mental models and the mapping between them
2. Be able to create conceptual designs from ecological, interaction, and emotional perspectives
3. Know what storyboards are and how to produce them
4. Understand the background aspects of embodied, ubiquitous, and situated interactions

8.1 INTRODUCTION

8.1.1 You Are Here

We begin each process chapter with a "you are here" picture of the chapter topic in the context of the overall Wheel lifecycle template; see Figure 8-1. This chapter is a continuation of design, which we started in Chapter 7 and will conclude in Chapter 9, for designing the new work practice and the new system.

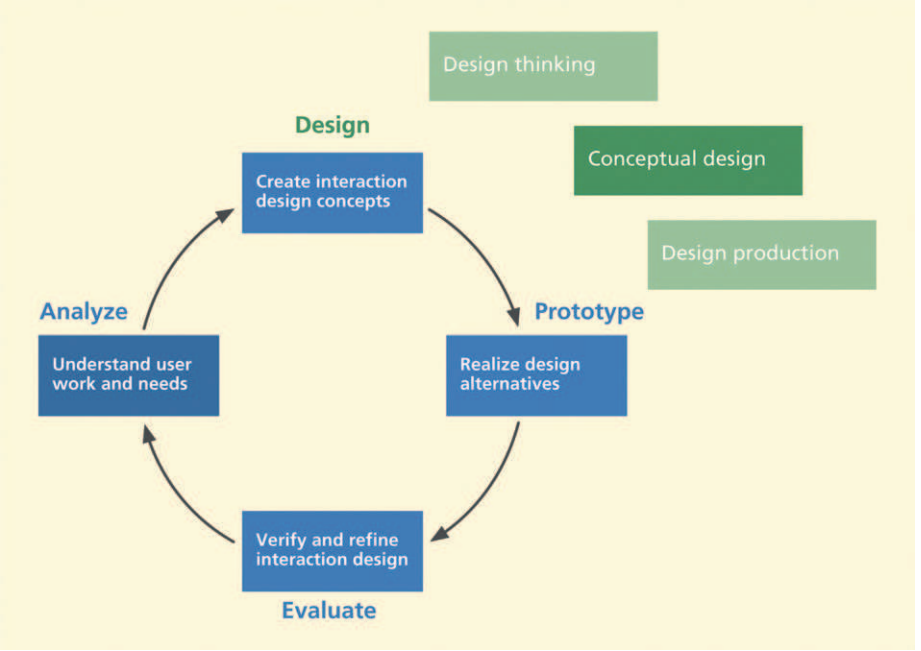
8.2 MENTAL MODELS

8.2.1 What Is a Mental Model?

According to Wikipedia.org, "a mental model is an explanation of someone's thought process about how something works in the real world." A designer's mental model is a vision of how a system works as held by the designer. A user's mental model is a description of how the system works, as held by the user. It is the job of conceptual design (coming up soon) to connect the two.

Figure 8-1

You are here; the second of three chapters on creating an interaction design in the context of the overall Wheel lifecycle template.



8.2.2 Designer's Mental Model

Sometimes called a conceptual model (Johnson & Henderson, 2002, p. 26), the designer's mental model is the designer's conceptualization of the envisioned system—what the system is, how it is organized, what it does, and how it works. If anyone should know these things, it is the designer who is creating the system. But it is not uncommon for designers to "design" a system without first forming and articulating a mental model.

The results can be a poorly focused design, not thought through from the start. Often such designs proceed in fits and starts and must be retraced and restarted when missing concepts are discovered along the way. The result of such a fuzzy start can be a fuzzy design that causes users to experience vagueness and misconceptions. It is difficult for users to establish a mental model of how the system works if the designer has never done the same.

Metaphor

A metaphors is an analogy used in design to communicate and explain unfamiliar concepts using familiar conventional knowledge. Metaphors control complexity by allowing users to adapt what they already know in learning how to use new system features.

As shown in Figure 8-2, the designer's mental model is created from what is learned in contextual inquiry and analysis and is transformed into design by ideation and sketching.

Johnson and Henderson (2002, p. 26) include metaphors, analogies, ontological structure, and mappings between those concepts and the task domain or work practice the design is intended to support. The closer the designer's mental model orientation is to the user's work domain and work

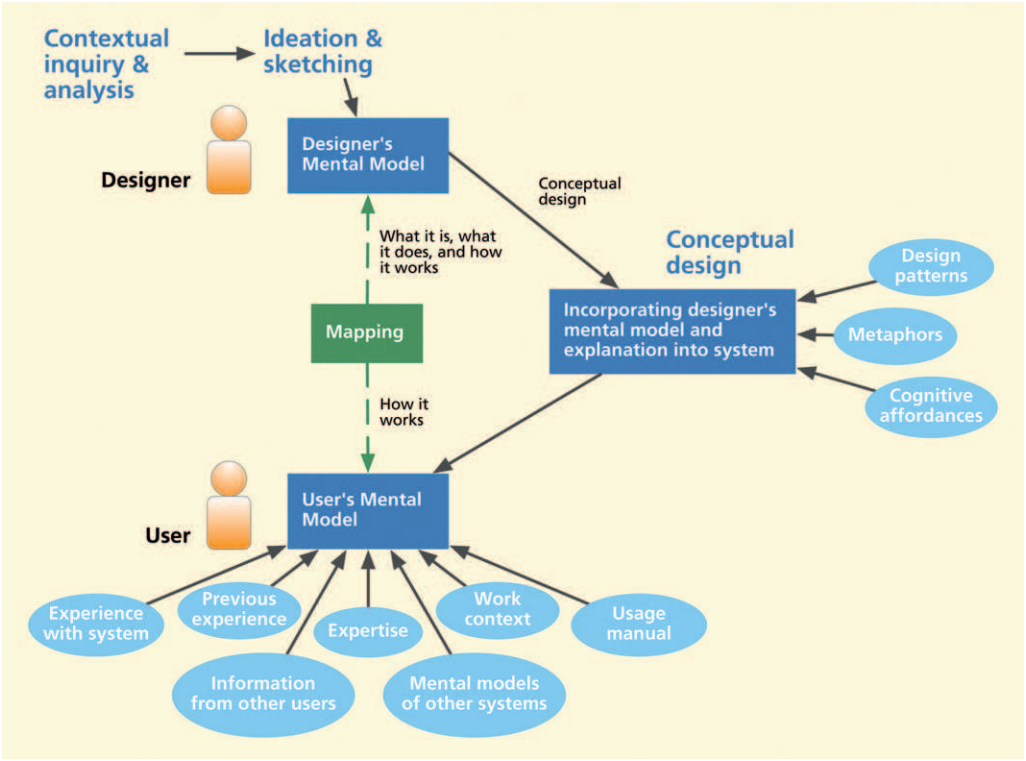


Figure 8-2
Mapping the designer's mental model to the user's mental model.

practice, the more likely users will internalize the model as their own. To paraphrase Johnson and Henderson's rule for relating the designer's mental model to the final design: if it is not in the designer's mental model, the system should not require users to be aware of it.

Designer's mental model in the ecological perspective: Describing what the system is, what it does, and how it works within its ecology

Mental models of a system can be expressed in any of the design perspectives of Chapter 7. In the ecological perspective, a designer's mental model is about how the system or product fits within its work context, in the flow of activities involving it and other parts of the broader system. In Norman's famous book, *The Design of Everyday Things*, he describes the use of thermostats (Norman, 1990, pp. 38–39) and how they work. Let us expand the explanation of thermostats to a description of what the system is and what it does from the perspective of its ecological setting.

Design Ontology

Design ontology is a description of all the objects and their relationships, users, user actions, tasks, everything surrounding the existence of a given aspect of a design.

First, we describe what it is by saying that a thermostat is part of a larger system, a heating (and/or cooling) system consisting of three major parts: a heat source, a heat distribution network, and a control unit, the latter being the thermostat and some other hidden circuitry. The heat source could be gas, electric, or wood burning, for example. The heat distribution network would use fans or air blowers to send heated or cooled air through hot air ducts or a pump would send heated or cooled water through subfloor pipes.

Next, we address what it does by noting that a thermostat is for controlling the temperature in a room or other space. It controls heating and cooling so that the temperature stays near a user-settable value—neither too hot or too cold—keeping people at a comfortable temperature.

Designer's mental model in the interaction perspective: Describing how users operate it

In the interaction perspective, a designer's mental model is a different view of an explanation of how things work; it is about how a user operates the system or product. It is a task-oriented view, including user intentions and sensory, cognitive, and physical user actions, as well as device behavior in response to these user actions.

In the thermostat example, a user can see two numerical temperature displays, either analog or digital. One value is for the current ambient temperature and the other is the setting for the target temperature. There will be a rotatable knob, slider, or other value-setting mechanism to set the desired target temperature. This covers the sensory and physical user actions for operating a thermostat. User cognition and proper formation of intentions with respect to user actions during thermostat operation, however, depend on understanding the usually hidden explanation of the behavior of a thermostat in response to the user's settings.

Most thermostats, as Norman explains (1990, pp. 38–39), are binary switches that are simply either on or off. When the sensed ambient temperature is below the target value, the thermostat turns the heat on. When the temperature then climbs to the target value, the thermostat turns the heat source off. It is, therefore, a false conceptualization, or false mental model, to believe that you can make a room warm up faster by turning the thermostat up higher.

The operator's manual for a particular furnace unit would probably say something to the effect that you turn it up and down to make it warmer or cooler, but would probably fall short of the full explanation of how a thermostat works. But the user is in the best position to form effective usage strategies,

connecting user actions with expected outcomes, if in possession of this knowledge of thermostat behavior.

There are at least two possible design approaches to thermostats, then. The first is the common design containing a display of the current temperature plus a knob to set the target temperature. A second design, which reveals the designer's mental model, might have a display unit that provides feedback messages such as "checking ambient temperature," "temperature lower than target; turning heat on," and "temperature at desired level; shutting off." This latter design might suffer from being more complex to produce and the added display might be a distraction to experienced users. However, this design approach does help project the designer's mental model through the system design to the user.

Designer's mental model in the emotional perspective: Describing intended emotional impact

In the emotional perspective, the mental model of a design it about the expected overarching emotional response. Regarding the thermostat example, it is difficult to get excited about the emotional aspects of thermostats, but perhaps the visual design, the physical design, how it fits in with the house décor, or the craftsmanship of its construction might offer a slight amount of passing pleasure.

8.2.3 User's Mental Model

A user's mental model is a conceptualization or internal explanation each user has built about how a particular system works. As Norman says (1990), it is a natural human response to an unfamiliar situation to begin building an explanatory model a piece at a time. We look for cause-and-effect relationships and form theories to explain what we observe and why, which then helps guide our behavior and actions in task performance.

As shown in Figure 8-2, each user's mental model is a product of many different inputs including, as Norman has often said, knowledge in the head and knowledge in the world. Knowledge in the head comes from mental models of other systems, user expertise, and previous experience. Knowledge in the world comes from other users, work context, shared cultural conventions, documentation, and the conceptual design of the system itself. This latter source of user knowledge is the responsibility of the system designer.

Few, if any, thermostat designs themselves carry any knowledge in the world, such as a cognitive affordance that conveys anything like Norman's explanation of a thermostat as a binary switch. As a result, thermostat users depend on

knowledge in the head, mostly from previous experience and shared conventions. Once you have used a thermostat and understand how it works, you pretty much understand all thermostats.

But sometimes mental models adapted from previous encounters with similar systems can work against learning to use a new system with a different conceptual design. Norman's binary switch explanation is accurate for almost every thermostat on the planet, but not for one in the heater of a mid-1960s Cadillac. In a fascinating departure from the norm, you could, in fact, speed up the heating system in this car, both the amount of heat and the fan speed, by setting the thermostat to a temperature higher than what you wanted in steady state.

Since cars were beginning to have more sophisticated (in this case, read more failure prone) electronics, why not put them to use? And they did. The output heat and fan speed were proportional to the difference between the ambient temperature and the thermostat setting. So, on a cold day, the heater would run wide open to produce as much heat as possible, but it would taper off its output as it approached the desired setting.

Lack of a correct user mental model can be the stuff of comedy curve balls, too. An example is the scene in the 1992 movie, *My Cousin Vinny*, where Marisa Tomei—as Vinny's fiancée, Mona Lisa Vito—tries to make a simple phone call. This fish-out-of-water scene pits a brash young woman from New York against a rotary dial telephone. You cannot help but reflect on the mismatch in the mapping between her mental model of touch-tone operation and the reality of old-fashioned rotary dials as she pokes vigorously at the numbers through the finger holes.

But, lest you dismiss her as a ditzy blond, we remind you that it was she who solved the case with her esoteric knowledge in the head, proving that the boys' 1964 Buick Skylark could not have left the two tire tracks found outside the convenience store because it did not have a limited-slip differential.

8.2.4 Mapping and the Role of Conceptual Design

The mapping in Figure 8-2 is an abstract and objective ideal transformation of the designer's mental model into the user's mental model (Norman, 1990, p. 23). As such the mapping is a yardstick against which to measure how closely the user's mental model matches the reality of the designer's mental model.

The conceptual design as it is manifest in the system is an implementation of this mapping and can be flawed or incomplete. A flawed conceptual design leads to a mismatch in the user's mental model. In reality, each user is likely to have a different mental model of the same system, and mental models can be incomplete and even incorrect in places.

8.3 CONCEPTUAL DESIGN

8.3.1 What Is a Conceptual Design?

A conceptual design is the part of an interaction design containing a theme, notion, or idea with the purpose of communicating a design vision about a system or product. A conceptual design is the manifestation of the designer's mental model within the system, as indicated in Figure 8-2. It is the part of the system design that brings the designer's mental model to life within the system. A conceptual design corresponds to what Norman calls the "system image" of the designer's mental model (Norman, 1990, pp. 16, 189–190), about which he makes the important point: this is the only way the designer and user can communicate.

Conceptual design is where you innovate and brainstorm to plant and first nurture the user experience seed. You can never iterate the design later to yield a good user experience if you do not get the conceptual part right up front. Conceptual design is where you establish the metaphor or the theme of the product—in a word, the concept.

8.3.2 Start with a Conceptual Design

Now that you have done your contextual inquiry and analysis, requirements, and modeling, as well as your ideation and sketching, how do you get started on design? Many designers start sketching out pretty screens, menu structures, and clever widgets.

But Johnson and Henderson (2002) will tell you to start with conceptual design before sketching any screen or user interface objects. As they put it, screen sketches are designs of "how the system *presents itself* to users. It is better to start by designing what the system *is* to them." Screen designs and widgets will come, but time and effort spent on interaction details can be wasted without a well-defined underlying conceptual structure. Norman (2008) puts it this way: "What people want is usable devices, *which translates into understandable ones*" (final emphasis ours).

To get started on conceptual design, gather the same team that did the ideation and sketching and synthesize all your ideation and sketching results into a high-level conceptualization of what the system or product is, how it fits within its ecology, and how it operates with users.

For most systems or products, especially domain-complex systems, the best way to start conceptual design is in the ecological perspective because that captures the system in its context. For product concepts where the emotional

impact is paramount, starting with that perspective is obvious. At other times the "invention" of an interaction technique like that of the iPod Classic scroll wheel might be the starting point for a solution looking for a problem and is best visualized in the interaction perspective.

8.3.3 Leverage Metaphors in Conceptual Design

One way to start formulating a conceptual design is by way of metaphors—analogies for communication and explanations of the unfamiliar using familiar conventional knowledge. This familiarity becomes the foundation underlying and pervading the rest of the interaction design.

What users already know about an existing system or existing phenomena can be adapted in learning how to use a new system (Carroll & Thomas, 1982). Use metaphors to control complexity of an interaction design, making it easier to learn and easier to use instead of trying to reduce the overall complexity (Carroll, Mack, & Kellogg, 1988).

One of the simple and oldest examples is the use of a typewriter metaphor in a word processing system. New users who are familiar with the knowledge, such as margin setting and tab setting in the typewriter domain, will already know much of what they need to know to use these features in the word processing domain.

Metaphors in the ecological perspective

Find a metaphor that can be used to describe the broader system structure. An example of a metaphor from the ecological perspective could be the description of iTunes as a mother ship for iPods, iPhones, and iPads. The intention is that all operations for adding, removing, or organizing media content, such as applications, music, or videos, are ultimately managed in iTunes and the results are synced to all devices through an umbilical connection.

Metaphors in the interaction perspective

An example of a metaphor in the interaction perspective is a calendar application in which user actions look and behave like writing on a real calendar. A more modern example is the metaphor of reading a book on an iPad. As the user moves a finger across the display to push the page aside, the display takes on the appearance of a real paper page turning. Most users find it comfortingly familiar.

Another great example of a metaphor in the interaction perspective can be found in the Time Machine feature on the Macintosh operating system. It is a backup feature where the user can take a "time machine" to go back to older

backups—by flying through time as guided by the user interface—to retrieve lost or accidentally deleted files.

One other example is the now pervasive desktop metaphor. When the idea of graphical user interfaces in personal computers became an economic feasibility, the designers at Xerox Parc were faced with an interesting interaction design challenge: How to communicate to the users, most of whom were going to see this kind of computer for the first time, how the interaction design works?

In response, they created the powerful "desktop" metaphor. The design leveraged the familiarity people had with how a desktop works: it has files, folders, a space where current work documents are placed, and a "trash can" where documents can be discarded (and later recovered, until the trash can itself is emptied). This analogy of a simple everyday desk was brilliant in its simplicity and made it possible to communicate the complexity of a brand new technology.

As critical components of a conceptual design, metaphors set the theme of how the design works, establishing an agreement between the designer's vision and the user's expectations. But metaphors, like any analogy, can break down when the existing knowledge and the new design do not match.

When a metaphor breaks down, it is a violation of this agreement. The famous criticism of the Macintosh platform's design of ejecting an external disk by dragging its icon into the trashcan is a well-known illustration of how a metaphor breakdown attracts attention. If Apple designers were faithful to the desktop metaphor, the system should probably discard an external disk, or at least delete its contents, when it is dragged and dropped onto the trashcan, instead of ejecting it.

Metaphors in the emotional perspective

An example of a metaphor from the emotional perspective is seen in advertising in *Backpacker* magazine of the Garmin handheld GPS as a hiking companion. In a play on words that ties the human value of self-identity with orienteering, Garmin uses the metaphor of companionship: "Find yourself, then get back." It highlights emotional qualities such as comfort, cozy familiarity, and companionship: "Like an old pair of boots and your favorite fleece, GPSMAP 62ST is the ideal hiking companion."

8.3.4 Conceptual Design from the Design Perspectives

Just as any other kind of design can be viewed from the three design perspectives of Chapter 7, so can conceptual design.

Conceptual design in the ecological perspective

The purpose of conceptual design from the ecological perspective is to communicate a design vision of how the system works as a black box within its environment. The ecological conceptual design perspective places your system or product in the role of interacting with other subsystems within a larger infrastructure.

As an example, Norman (2009) cites the Amazon Kindle™—a good example of a product designed to operate within an infrastructure. The product is for reading books, magazines, or any textual material. You do not need a computer to download or use it; the device can live as its own independent ecology. Browsing, buying, and downloading books and more is a pleasurable flow of activity. The Kindle is mobile, self-sufficient, and works synergistically with an existing Amazon account to keep track of the books you have bought through Amazon.com. It connects to its ecology through the Internet for downloading and sharing books and other documents. Each Kindle has its own email address so that you and others can send lots of materials in lots of formats to it for later reading.

As discussed previously, the way that iPods and iTunes work together is another example of conceptual design in the ecological perspective. Norman calls this designing an infrastructure rather than designing just an application. Within this ecosystem, iTunes manages all your data. iTunes is the overall organizer through which you buy and download all content. It is also where you create all your playlists, categories, photo albums, and so on. Furthermore, it is in iTunes that you decide what parts of your data you want on your "peripherals," such as an iPod, iPad, or iPhone. When you connect your iDevice to the computer and synchronize it, iTunes will bring it up to date, including an installation of the latest version of the software as needed.

Usability of an Ecology of Devices: A Personal Information Ecosystem

Manuel A. Pérez-Quiñones, Department of Computer Science, Virginia Tech

The world of ubiquitous computing imagined by Mark Weiser (1991) is upon us. The computational power of small devices is enabling new uses of computing away from the desktop or office. Networking and communication abilities of devices make it possible to use computing in mobile settings. Storage and display improvements make difficult

tasks now possible on small devices. For example, one can do photo and video editing on an iPhone. The "cloud" is tying all of these together and providing access to computing and information anytime, anywhere.

In this new environment, the biggest challenge for usability engineers is that all of these devices are used together to accomplish user's information needs and goals. Whereas before we had tools dedicated to particular tasks (e.g., email programs), now we have a set of devices, each with a set of tools to support the same tasks. The usability of these tasks must be evaluated as a collection of devices working together, not as the sum of the usability of individual tools. Some tasks, on the surface, can be done on any of our many devices. Take email, for example. You can read, reply, forward, and delete emails in your phone, tablet device, laptop, desktop, game console, or even TV or entertainment center. However, managing email sometimes entails more than that. Once you get to filing and refinding previous email messages, the tasks gets very complicated on some of these devices. And opening some attachments might not be possible in other devices. Also, even though we have connectivity to talk to anyone in the world, you do not quite have enough connectivity to print an email remotely at home or at the office. The result is that not all devices support all the tasks required to accomplish our work, but the collection of devices together do, while allowing mobility and 24/7 access to information.

The challenge comes on how to evaluate a system of coordinated device usage that spans multiple manufacturers, multiple communication capabilities, and multiple types of activities. The experience of using (and configuring and managing) multiple devices together is very different than using only one device. As a matter of fact, the usability of just one device is barely a minimum fit for it to work within the rest of devices used in our day-to-day information management. Furthermore, the plethora of devices creates a combinatorial explosion of device choices that make assessing the usability of the devices together practically impossible.

Part of the problem is that we lack a way to understand and study this collection of devices. To alleviate this need, we have proposed a framework, called a personal information ecosystem (PIE) (Pérez-Quiñones et al., 2008), that at least helps us characterize different ecologies that emerge for information management. The idea of ecosystems in information technology is not new, but our approach is most similar to Spinuzzi's (2001) ecologies of genre. Spinuzzi argues that usability is not an attribute of a single product or artifact, but that instead it is best studied across the entire ecosystem used in an activity. His approach borrows ideas from distributed cognition and activity theory.

At the heart of the ecology of devices is an information flow that is at its optimum point (i.e., equilibrium) when the user is exerting no extra effort to accomplish his/her tasks. At equilibrium, the user rarely needs to think of the devices, the data format, or the commands to move information to and from devices. This equilibrium, however, is disrupted easily by many situations: introduction of a new device, disruption in service (wifi out of range), changes in infrastructure, incompatibility between programs, etc. It is often quite a challenge to have all of your devices working together to reach this equilibrium. The usability of the ecosystem depends more on the equilibrium and ease of information flow than on the individual usability of each device.

However, having a terminology and understanding the relationships between devices are only the beginning. I would claim that designing and assessing user experience within an ecology of devices is what Rittel (1972) calls a "wicked problem." A wicked problem, according to Rittel, is a problem that by its complexity and nature cannot have a definitive formulation. He even states that a formulation of the problem itself corresponds to a particular solution of the problem. Often, wicked problems have no particular solution, instead we judge a solution

as good or bad. We often cannot even test a solution to a wicked problem, we can only indicate to a degree to which a given solution is good. Finally, in wicked problems, according to Rittel, there are many explanations for the same discrepancy and there is no way to test which of these explanations is the best one. In general, every wicked problem can be considered a symptom of another problem.

Why is designing and assessing usability of an ecology a wicked problem? First, different devices are often designed by different companies. We do not really know which particular combination of devices a given user will own. Evaluating all combinations is prohibitively expensive, and expecting one company to provide all the devices is not ideal either, as monopolies tend to stifle innovation. As a result, the user is stuck in an environment that can at best provide a local optimum—"if you use this device with this other device, then your email will work ok."

Second, while some problems are addressed easily by careful design of system architecture, eventually new uses emerge that were not anticipated by the designers. For example, if a user is using IMAP as the server protocol for his/her email, then all devices are "current" with each other as the information about her/his email is stored in a central location. But even this careful design of network protocols and systems architecture cannot account for all the uses that evolve over time. The email address autocompletion and the signature that appears at the bottom of your email are both attributes of the clients and are not in the IMAP protocol. Thus, a solution based on standards can only support agreed common tasks from the past but does not support emergent behavior.

Third, the adoption of a new device into the ecology often breaks other parts that were already working effectively. As a result, whatever effort has gone into solving a workflow problem is lost when a different combination of devices is present. For example, I use an Apple MacBook Pro as my main computer, an iPad for most of my home use, and an Android phone for my communication needs. At times, finding a good workflow for these three devices is a challenge. I have settled on using GMail and Google Calendar in all three devices because there is excellent support for all three. But other genres are not as well supported. Task management, for example, is one where I currently do not have a good solution that works in my phone, the most recent addition to my PIE. New devices upset the equilibrium of the ecosystem; the problem that I am addressing (task management) is a symptom of another problem I introduced.

Fourth, the impact of the changes in an ecosystem is highly personalized. I know users whose email management and information practices improved when they obtained a smartphone. For them, most of their email traffic was short and for the purpose of coordinating meetings or upcoming events. Introduction of a smartphone allowed them to be more effective in their email communication. For me, for example, the impact was the opposite. As most knowledge workers, I do a lot of work over email with discussions and document exchanges. The result is that I tag my email and file messages extensively. But because my phone and tablet device provide poor support for filing messages, I now leave more messages in my inbox to be processed when I am back on my laptop. Before I added my smartphone to my ecosystem, my inbox regularly contained 20 messages. Now, my inbox has pending tasks from when I was mobile. The result is that I have 50 to 60 messages regularly in my inbox. Returning to my laptop now requires that I "catch-up" on work that I did while mobile. The impact of adding a smartphone has been negative to me, in some respects, whereas for other users it had a positive effect.

Finally, a suitable solution to a personal ecosystem is one that depends on the user doing some work as a designer of his or her own information flow. Users have to be able to observe their use, identify their own inefficiencies, propose solutions, and design workflows that implement those solutions. Needless to say, not every user has the skills

to be a designer and to even be able to self-assess where their information flow is disrupted. Spinuzzi (2001) discusses this point using the Bødker (1991) concept of breakdowns. Paraphrasing Spinuzzi, breakdowns are points at which a person realizes that his or her information flow is not working as expected and thus the person must devote attention to his or her tools/ecosystem instead of his or her work. Typically this is what a usability engineer would consider a usability problem, but in the context of a PIE, this problem is so deeply embedded in the particular combination of devices, user tasks, and user information flows that it is practically impossible for a usability engineer to identify this breakdown. We are left with the user as a designer as the only option of improving the usability of a PIE.

As usability engineers, we face a big challenge on how to study, design, and evaluate user experience of personal information ecosystem that have emerged in today's ubiquitous environments.

References

Bødker, S. (1991). *Through the Interface: A Human Activity Approach to User Interface Design.* Hillsdale, New Jersey: Erlbaum.

Pérez-Quiñones, M. A., Tungare, M., Pyla, P. S., & Harrison, S. (2008). Personal Information Ecosystems: Design Concerns for Net-Enabled Devices. In *Proceedings of Latin American-WEB'2008 Conference*, (pp. 3–11). October 28–30, Vila Velha, Espírito Santo, Brasil.

Rittel, H. (1972). On the planning crisis: Systems Analysis of the 'first and Second Generations'. *Bedriftskonomen*, *8*, 390–396.

Spinuzzi, C. (2001). Grappling with Distributed Usability: A Cultural-Historical Examination of Documentation Genres over Four Decades. *Journal of Technical Writing and Communication*, *31*(1), 41–59.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 94–100, September.

Conceptual design in the interaction perspective

The conceptual design from the interaction perspective is used to communicate a design vision of how the user operates the system. A good example of conceptual design from an interaction perspective is the Mac Time Machine backup feature discussed previously. Once that metaphor is established, the interaction design can be fleshed out to leverage it.

The designers of this feature use smooth animation through space to represent traveling through the different points in time where the user made backups. When the user selects a backup copy from a particular time in the past, the system lets the user browse through the files from that date. Any files from that backup can be selected and they "travel through time" to the present, thereby recovering the lost files.

As an example of designers leveraging the familiarity of conceptual designs from known applications to new ones, consider a well-known application such as Microsoft Outlook. People are familiar with the navigation bar on the left-hand side, list view at the top right-hand side, and a preview of the

selected item below the list. When designers use that same idea in the conceptual design of a new application, the familiarity carries over.

Conceptual design in the emotional perspective

Conceptual design from the emotional perspective is used to communicate a vision of how the design elements will evoke emotional impact in users. Returning to the car example, the design concept could be about jaw-dropping performance and how your heart skips a beat when you see its aerodynamic form or it could be about fun and being independent from the crowd. Ask any MINI driver about what their MINI means to them.

In Figure 8-3 we summarize conceptual design in the three perspectives.

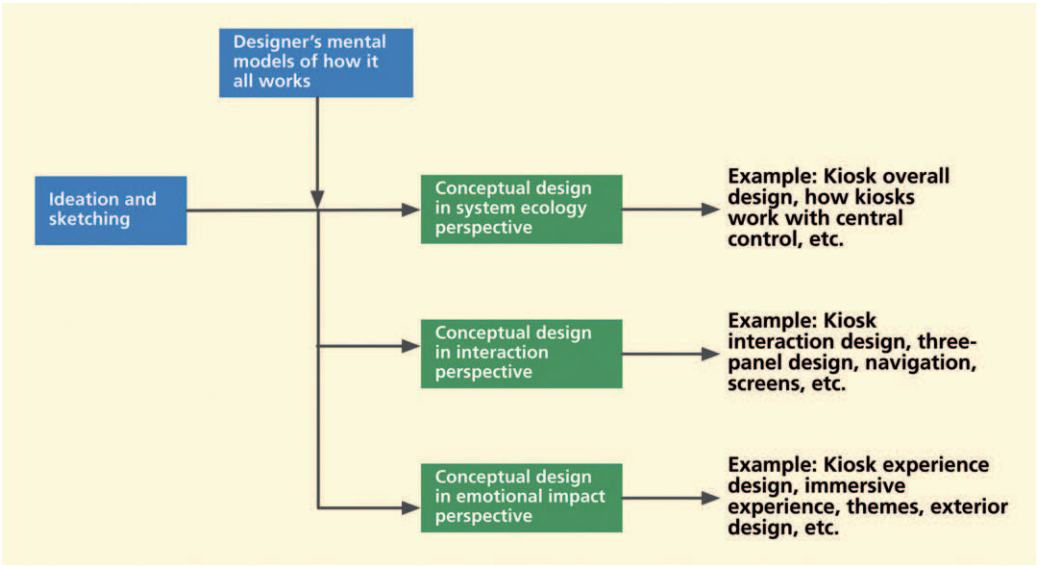


Figure 8-3

Designer workflow and connections among the three conceptual design perspectives.

Example: Conceptual Design for the Ticket Kiosk System

There is a strong commonly held perception of a ticket kiosk that includes a box on a pedestal and a touchscreen with colorful displays showing choices of events. If you give an assignment to a team of students, even most HCI students, to come up with a conceptual design of a ticket kiosk in 30 minutes, 9 times out of 10 you will get something like this.

But if you teach them to approach it with design thinking and ideation, they can come up with amazingly creative and varied results.

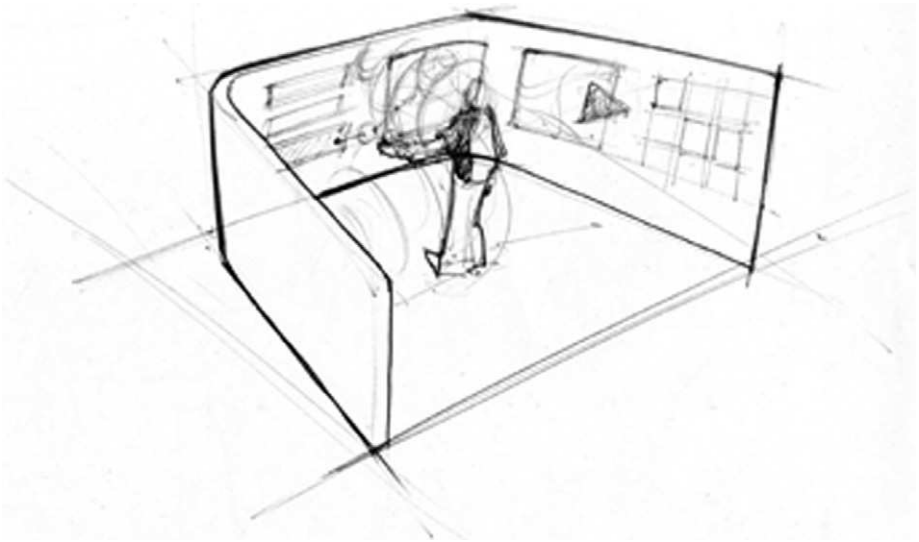


Figure 8-4

Part of a conceptual design showing immersion in the emotional perspective (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

In our ideation about the Ticket Kiosk System, someone mentioned making it an immersive experience. That triggered more ideas and sketches on how to make it immersive, until we came up with a three-panel overall design. In Figure 8-4 we show this part of a conceptual design for the Ticket Kiosk System showing immersion in the emotional perspective.

Here is a brief description of the concept, in outline form.

- The center screen is the interaction area, where immersion and ticket-buying action occur.
- The left-hand screen contains available options or possible next steps; for example, this screen might provide a listing of all required steps to complete a transaction, including letting user access these steps out of sequence.
- The right-hand screen contains contextual support, such as interaction history and related actions; for example, this screen might provide a summary of the current transaction so far and related information such as reviews and ratings.
- The way that the three panels lay out context as a memory support and for consistent use is a kind of human-as-information-processor concept.
- Using the sequence of panels to represent the task flow is a kind of engineering concept.
- Each next step selection from the left-hand panel puts the user in a new kind of immersion in the center screen, and the previous immersion situation becomes part of the interaction history on the right-hand panel.

- Addressing privacy and enhancing the impression of immersion: When the ticket buyer steps in, rounded shields made of classy materials gently wrap around. An "Occupied" sign glows on the outside. The inside of the two rounded half-shells of the shield become the left-hand-side and right-hand-side interaction panels.

In Figure 8-5 we show ideas from an early conceptual design for the Ticket Kiosk System from the ecological perspective.

In Figure 8-6 we show ideas from an ecological conceptual design for the Ticket Kiosk System focusing on a feature for a smart ticket to guide users to seating.

In Figure 8-7 we show ecological conceptual design ideas for the Ticket Kiosk System focusing on a feature showing communication connection with a smartphone. You can have a virtual ticket sent from a kiosk to your mobile device and use that to enter the event.

Exercise

See Exercise 8-1, Conceptual Design for Your System

In Figure 8-8 we show ecological conceptual design ideas for the Ticket Kiosk System focusing on the features for communicating and social networking.

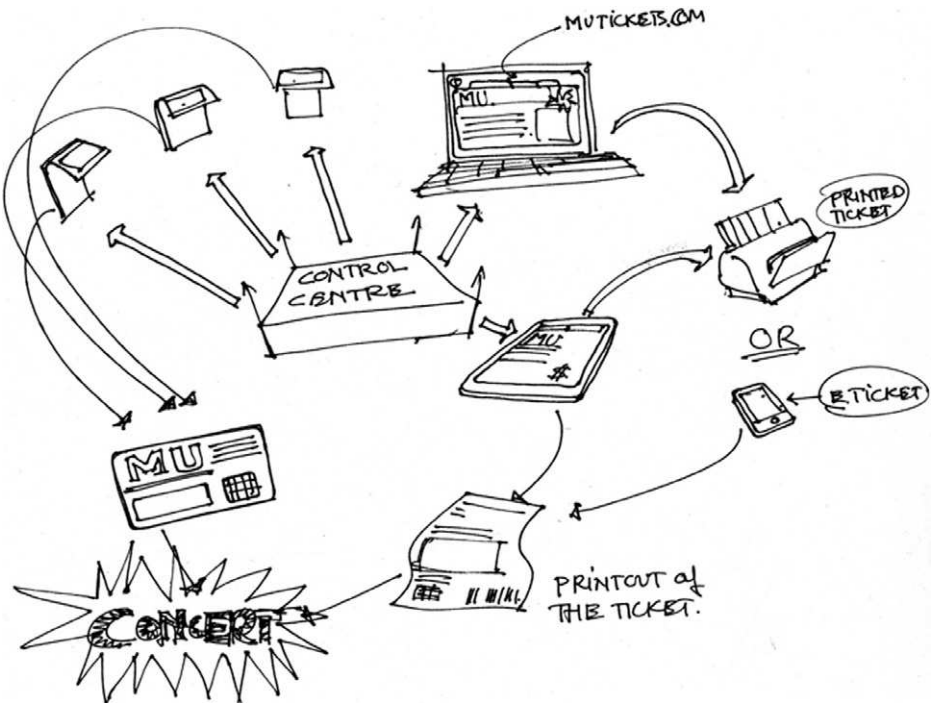


Figure 8-5

Early conceptual design ideas from the ecological perspective (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

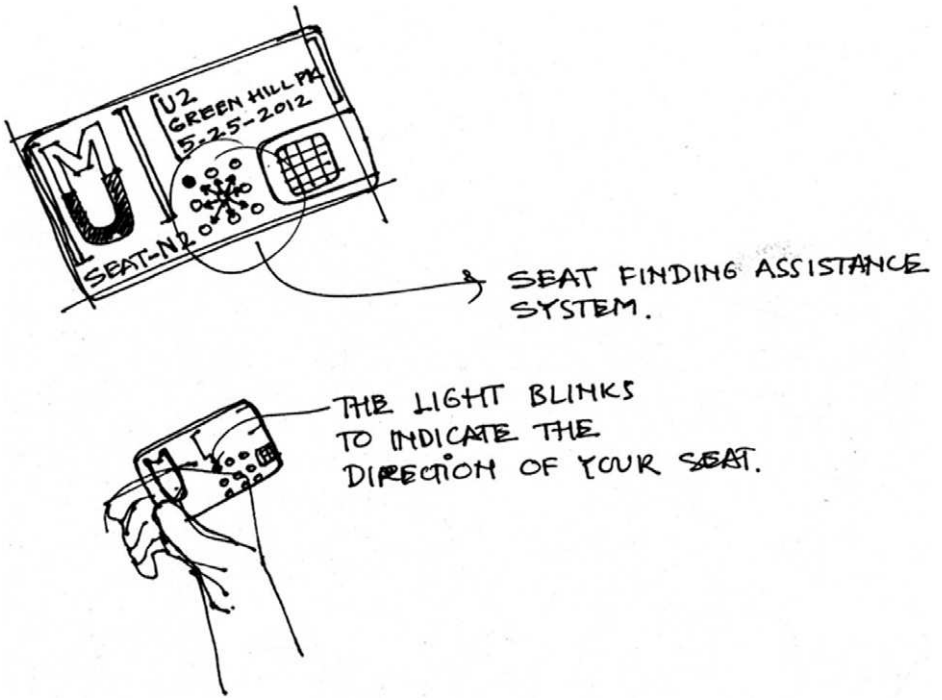


Figure 8-6

Ecological conceptual design ideas focusing on a feature for a smart ticket to guide users to seating (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

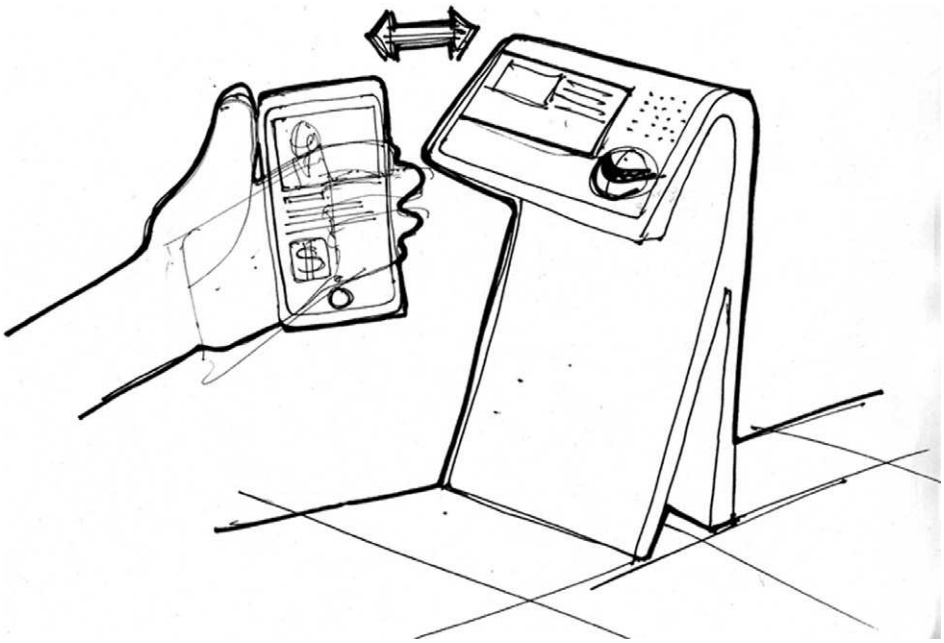


Figure 8-7

Ecological conceptual design ideas focusing on a feature showing communication connection with a smartphone (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

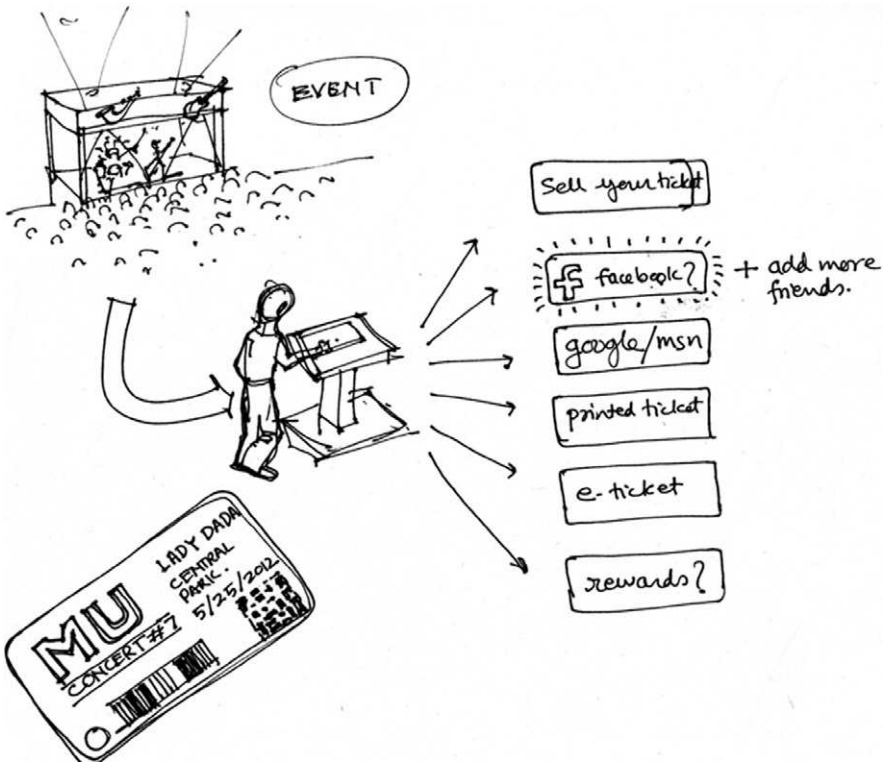


Figure 8-8
Ecological conceptual design ideas focusing on the features for communicating and social networking (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

In Figure 8-9 we show part of a conceptual design for the Ticket Kiosk System in the interaction perspective.

8.4 STORYBOARDS

8.4.1 What Are Storyboards?

A storyboard is a sequence of visual "frames" illustrating the interplay between a user and an envisioned system. Storyboards bring the design to life in graphical "clips," freeze-frame sketches of stories of how people will work with the system. This narrative description can come in many forms and at different levels.

Storyboards for representing interaction sequence designs are like visual scenario sketches, envisioned interaction design solutions. A storyboard might be thought of as a "comic-book" style illustration of a scenario, with actors, screens, interaction, and dialogue showing sequences of flow from frame to frame.

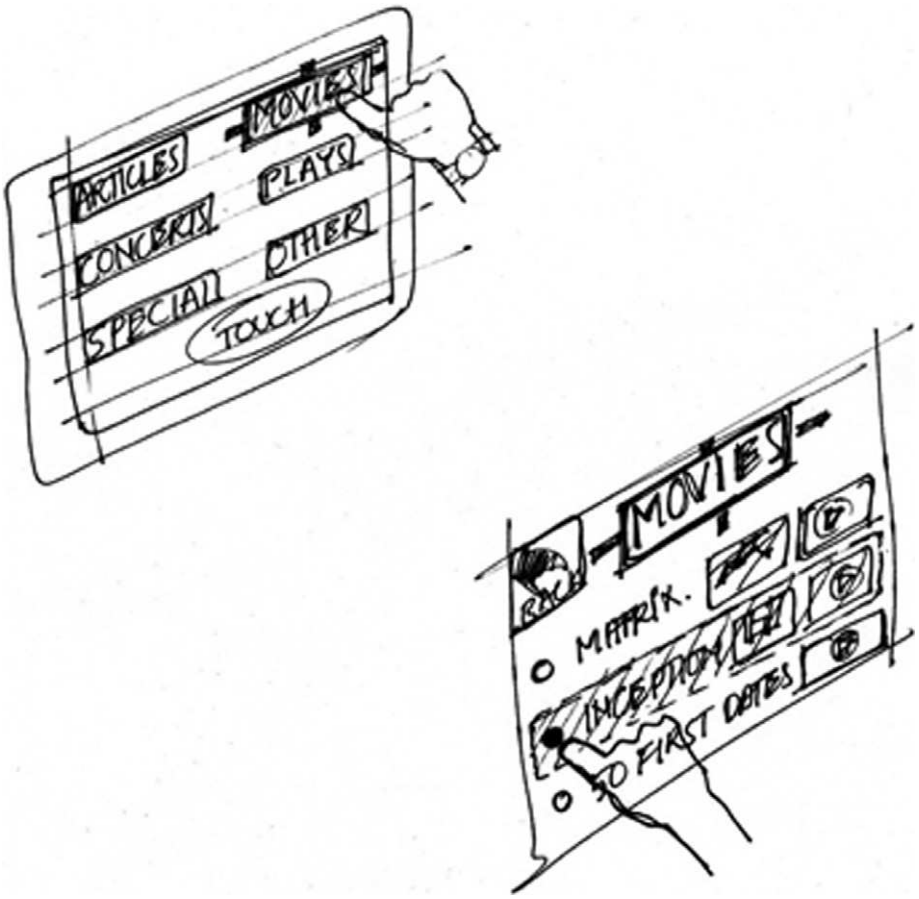


Figure 8-9

Part of a conceptual design in the interaction perspective (sketch courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

8.4.2 Making Storyboards to Cover All Design Perspectives

From your ideation and sketches, select the most promising ideas for each of the three perspectives. Create illustrated sequences that show each of these ideas in a narrative style.

Include things like these in your storyboards:

- Hand-sketched pictures annotated with a few words
- All the work practice that is part of the task, not just interaction with the system, for example, include telephone conversations with agents or roles outside the system
- Sketches of devices and screens
- Any connections with system internals, for example, flow to and from a database
- Physical user actions

- Cognitive user actions in "thought balloons"
- Extra-system activities, such as talking with a friend about what ticket to buy

For the ecological perspective, illustrate high-level interplay among human users, the system as a whole, and the surrounding context. Look at the envisioned flow model for how usage activities fit into the overall flow. Look in the envisioned social model for concerns and issues associated with the usage in context and show them as user "thought bubbles."

As always in the ecological perspective, view the system as a black box to illustrate the potential of the system in a context where it solves particular problems. To do this, you might show a device in the hands of a user and connect its usage to the context. As an example, you might show how a handheld device could be used while waiting for a flight in an airport.

In the interaction perspective, show screens, user actions, transitions, and user reactions. You might still show the user, but now it is in the context of user thoughts, intentions, and actions upon user interface objects in operating the device. Here is where you get down to concrete task details. Select key tasks from the HTI, design scenarios, and task-related models to feature in your interaction perspective storyboards.

Use storyboards in the emotional perspective to illustrate deeper user experience phenomena such as fun, joy, and aesthetics. Find ways to show the experience itself—remember the excitement of the mountain bike example from Buxton (Chapter 1).

Example: Ticket Kiosk System Storyboard Sketches in the Ecological Perspective

See Figure 8-10 for an example of a sequence of sketches as a storyboard depicting a sequence using a design in the ecological perspective.

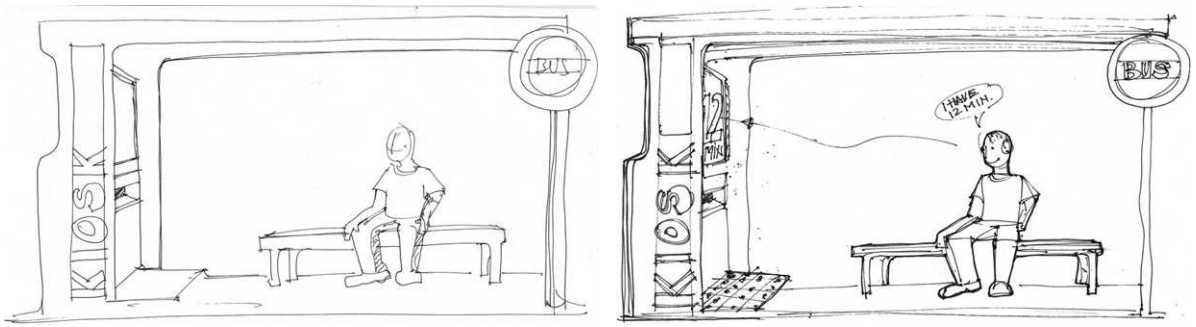


Figure 8-10
Example of a sequence of sketches as a storyboard in the ecological perspective (sketches courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

Continued

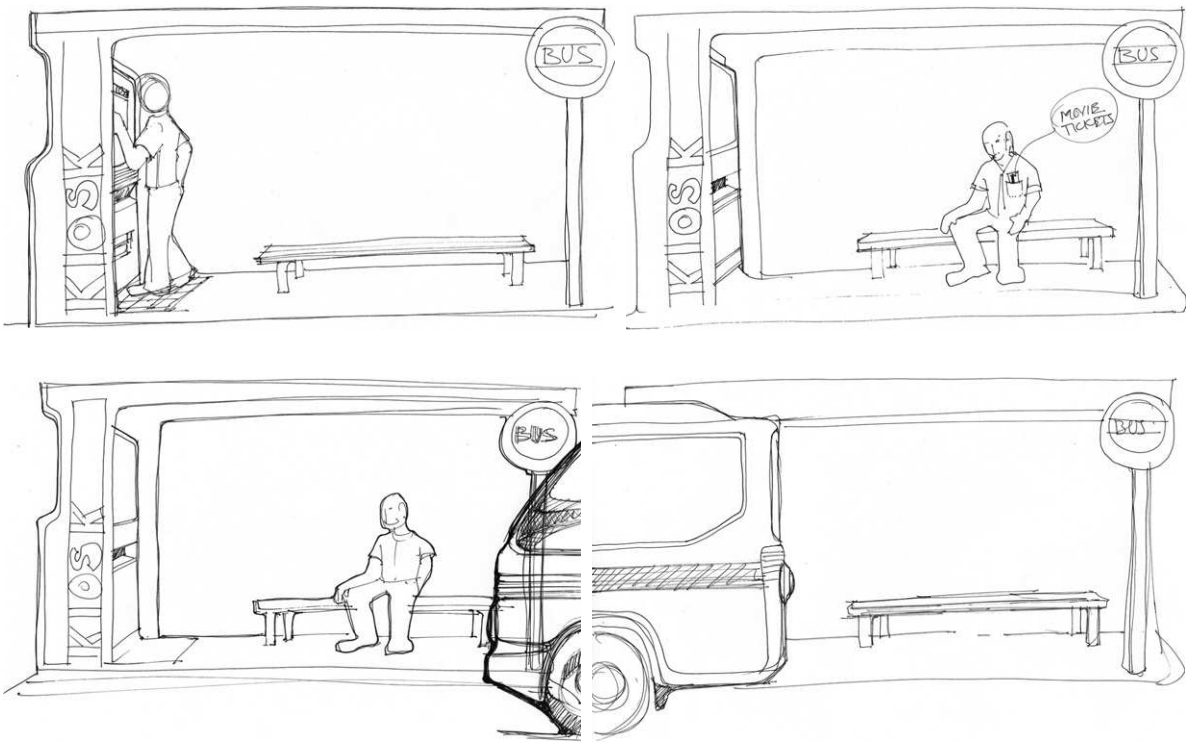


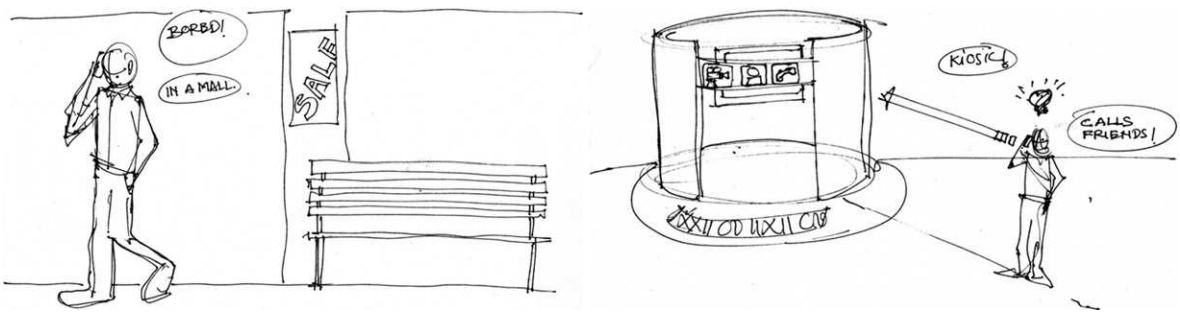
Figure 8.10, cont'd

Example: More Ticket Kiosk System Storyboard Sketches in the Ecological Perspective

In Figure 8-11 we show part of a different Ticket Kiosk System storyboard in the ecological perspective.

Figure 8-11

Part of a different Ticket Kiosk System storyboard in the ecological perspective (sketches courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).



Example: Ticket Kiosk System Storyboard Sketches in the Interaction Perspective

The following is one possible scenario that came out of an ideation session for an interaction sequence for a town resident buying a concert ticket from the Ticket Kiosk System. This example is a good illustration of the breadth we intend for the scope of the term "interaction," including a person walking with respect to the kiosk, radio-frequency identification at a distance, and audio sounds being made and heard. This scenario uses the three-screen kiosk design, where LS = left-hand screen, CS = center screen, RS = right-hand screen, and SS = surround sound.

- Ticket buyer walks up to the kiosk
- Sensor detects and starts the immersive protocol
- Provides "Occupied" sign on the wrap-around case
- Detects people with MU passports
- Greets buyer and asks for PIN
- [CS] Shows recommendations and most popular current offering based on buyer's category
- [RS] Shows buyer's profile if one exists on MU system
- [LS] Lists options such as browse events, buy tickets, and search
- [CS] Buyer selects "Boston Symphony at Burruss Hall" from the recommendations
- [RS] "Boston Symphony at Burruss Hall" title and information and images
- [SS] Plays music from that symphony
- [CS] Plays simulated/animated/video of Boston Symphony in a venue that looks like Burruss Hall. Shows "pick date and time"
- [LS] Choices, pick date and time, go back, exit.
- [CS] Buyer selects "pick date and time" option
- [CS] A calendar with "Boston Symphony at Burruss Hall" is highlighted, with other known events and activities with clickable dates.
- [CS] Buyer selects date from the month view of calendar (can be changed to week)
- [RS] The entire context selected so far, including date
- [CS] A day view with times, such as Matinee or evening. The rest of the slots in the day show related events such as wine tasting or special dinner events.
- [LS] Options for making reservations at these special events
- [CS] Buyer selects a time
- [RS] Selected time
- [CS] Available seating chart with names for sections/categories aggregate number of available seats per each section

- [LS] Categories of tickets and prices
- [CS] Buyer selects category/section
- [RS] Updates context
- [CS] Immerses user from a perspective of that section. Expands that section to show individual available seats. Has a call to action "Click on open seats to select" and an option to specify number of seats.
- [LS] Options to go back to see all sections or exit
- [CS] Buyer selects one or more seats by touching on available slots. A message appears "Touch another seat to add to selection or touch selected seat to unselect."
- [CS] Clicks on "Seat selection completed"
- [RS] Updates context
- [CS] Shows payment options and a virtual representation of selected tickets
- [LS] Provides options with discounts, coupons, sign up for mailing lists, etc.
- [CS] Buyer selects a payment option
- [CS] Provided with a prompt to put credit card in slot
- [CS] Animates to show a representation of the card on screen
- [CS] Buyer completes payment
- [LS] Options for related events, happy hour dinner reservations, etc. These are contextualized to the event they just bought the tickets just now.
- [CS] Animates with tickets and CC coming back out of their respective slots

In Figure 8-12 we have shown sample sketches for a similar storyboard.

8.4.3 Importance of Between-Frame Transitions

Storyboard frames show individual states as static screenshots. Through a series of such snapshots, storyboards are used to show the progression of interaction over time. However, the important part of cartoons (and, by the same token, storyboards) is the space between the frames (Buxton, 2007b). The frames do not reveal how the transitions are made.

For cartoons, it is part of the appeal that this is left to the imagination, but in storyboards for design, the dynamics of interaction in these transitions are where the user experience lives and the actions between frames should be part of what is sketched. The transitions are where the cognitive affordances in your design earn their keep, where most problems for users exist, and where the challenges lie for designers.

Cognitive Affordance

A cognitive affordance is a design feature that helps users with their cognitive actions: thinking, deciding, learning, remembering, and knowing about things.

We can augment the value of our storyboards greatly to inform design by showing the circumstances that lead to and cause the transitions and the context,

situation, or location of those actions. These include user thoughts, phrasing, gestures, reactions, expressions, and other experiential aspects of interaction. Is the screen difficult to see? Is the user too busy with other things to pay attention to the screen? Does a phone call lead to a different interaction sequence?

Exercise

See Exercise 8-2, Storyboard for Your System

In Figure 8-13 we show a transition frame with a user thought bubble explaining the change between the two adjacent state frames.

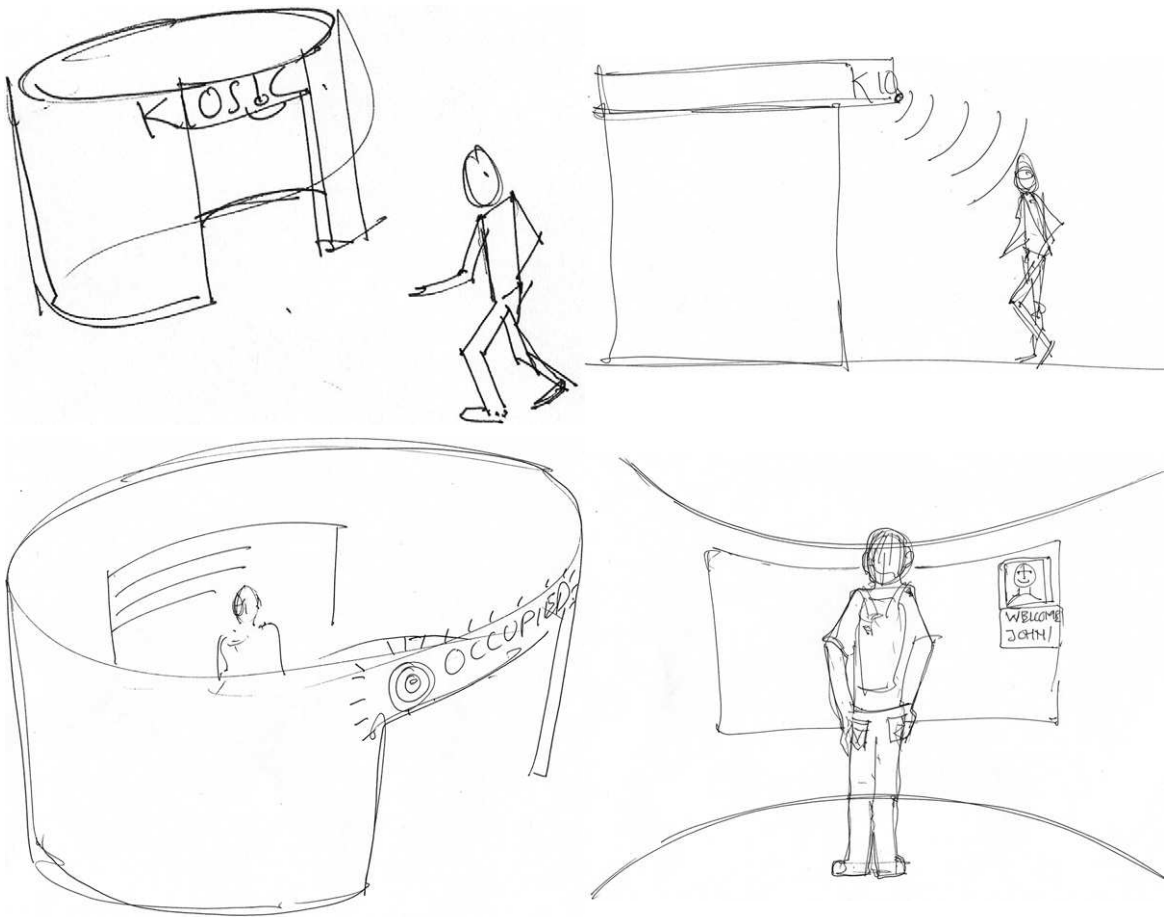


Figure 8-12

Sample sketches for a similar concert ticket purchase storyboard in the interaction perspective (sketches courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

Continued

Figure 8.12, cont'd

Figure 8.12, cont'd

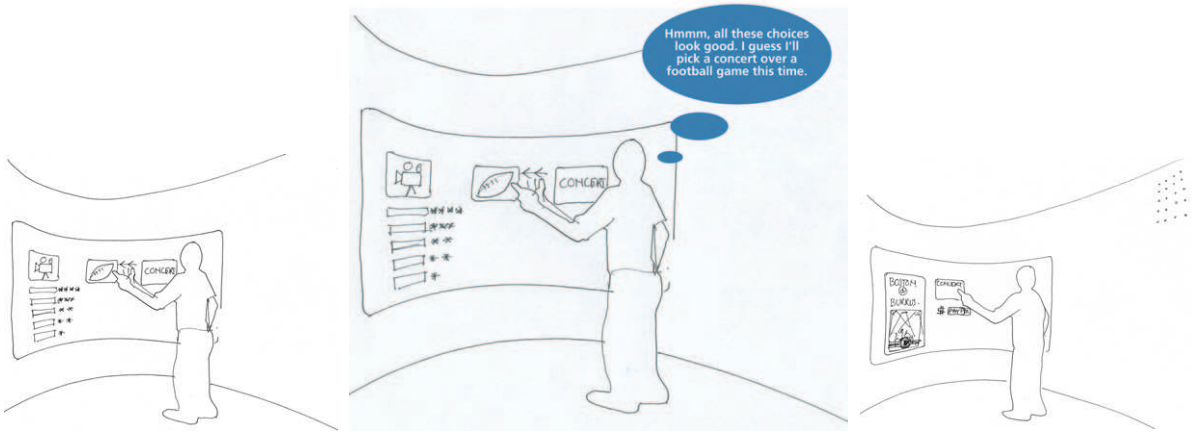


Figure 8-13

Storyboard transition frame with thought bubble explaining state change (sketches courtesy of Akshay Sharma, Virginia Tech Department of Industrial Design).

8.5 DESIGN INFLUENCING USER BEHAVIOR

Beale (2007) introduces the interesting concept of slanty design. "Slanty design is an approach that extends user-centered design by focusing on the things people should (and should not) be able to do with the product(s) behind the design." Design is a conversation between designers and users about both desired and undesired usage outcomes. But user-centered design, for example, using contextual inquiry and analysis, is grounded in the user's current

behavior, which is not always optimal. Sometimes, it is desirable to change, or even control, the user's behavior.

The idea is to make a design that works best for all users taken together and for the enterprise at large within the ecological perspective. This can work against what an individual user wants. In essence, it is about controlling user behavior through designs that attenuate usability from the individual user's interaction perspective, making it difficult to do things not in the interest of other users or the enterprise in the ecological perspective, but still allowing the individual users to accomplish the necessary basic functionality and tasks.

One example is sloped reading desks in a library, which still allow reading but make it difficult to place food or drink on the desk or, worse, on the documents. Beale's similar example in the domain of airport baggage claims is marvelously simple and effective. People stand next to the baggage conveyor belt and many people even bring their carts with them. This behavior increases usability of the system for them because the best ease of use occurs when you can just pluck the baggage from the belt directly onto the cart.

However, crowds of people and carts cause congestion, reducing accessibility and usability of other users with similar needs. Signs politely requesting users to remain away from the belt except at the moment of luggage retrieval are regrettably ineffective. A slanty design for the baggage carousel, however, solves the problem nicely. In this case, it involves something that is physically slanty; the surrounding floor slopes down away from the baggage carousel.

This interferes with bringing carts close to the belt and significantly reduces the comfort of people standing near the belt, thus reducing individual usability by forcing people to remain away from the carousel and then make a dash for the bags when they arrive within grasping distance. But it works best overall for everyone in the ecological perspective. Slanty design includes evaluation to eliminate unforeseen and unwanted side effects.

There are other ways that interaction design can influence user behavior. For example, a particular device might change reading habits. The Amazon Kindle device, because of its mobility and connectedness, makes it possible for users to access and read their favorite books in many different environments. As another example, interaction design can influence users to be "green" in their everyday activities. Imagine devices that detect the proximity of the user, shutting themselves down when the user is no longer there, to conserve power.

The Green Machine User-Experience Design: An Innovative Approach to Persuading People to Save Energy with a Mobile Device That Combines Smart Grid Information Design Plus Persuasion Design

Aaron Marcus, President, and Principal Designer/Analyst, Aaron Marcus and Associates, Inc. (AM+A)

In past decades, electric meters in homes and businesses were humble devices viewed primarily by utility company service technicians. Smart Grid developments to conserve energy catapult energy data into the forefront of high-technology innovation through information visualization, social media, education, search engines, and even games and entertainment. Many new techniques of social media are transforming society and might incorporate Smart Grid data. These techniques include the following:

- **Communication:** Blogs, microblogging, social networking, soc net aggregation, event logs/tracking
- **Collaboration:** wikis, social bookmarking (social tagging), social news, opinions, Yelp
- **Multimedia:** photo/video sharing, livecasting, audio/music sharing
- **Reviews and opinions:** product/business reviews, community Q+As
- **Entertainment:** platforms, virtual worlds, game sharing

Prototypes of what might arise are to be found in many places around the Internet. As good as these developments are, they do not go far enough. Just showing people information is good, but not sufficient. What seems to be missing is persuasion.

We believe that one of the most effective ways in which to reach people is to consider mobile devices, in use by more than three billion people worldwide. Our Green Machine mobile application prototype seeks to persuade people to save energy.

Research has shown that with feedback, people can achieve a 10% energy-consumption reduction without a significant lifestyle change. In the United States, this amount is significant, equal to the total energy provided by wind and solar resources, about 113.9 billion kwh/year. President Obama allocated more than $4 billion in 2010 Smart Grid funding to help change the context of energy monitoring and usage. Most of the Smart Grid software development has focused on desktop personal computer applications. Relatively few have taken the approach of exploring the use of mobile devices, although an increasing number are being deployed.

For our Green Machine project, we selected a home-consumer context to demonstrate in an easy-to-understand example how information design could be merged with persuasion design to change users' behavior. The same principles can be reapplied to the business context, to electric vehicle usage, and to many other contexts. For our use scenario, we assumed typical personas, or user profiles: mom, dad, and the children, who might wish to see their home energy use status and engage with the social and information options available on their mobile devices.

We incorporated five steps of behavior-changing process: increasing frequency of use of sustainability tools, motivating people to reduce energy consumption, teaching them how to reduce energy consumption, persuading them to make short-term changes, and persuading them to make long-term changes in their behavior. This process included, for example, the following techniques: rewards, using user-centered design, motivating people via views into the future, motivating them through games, providing tips to help people get started and to learn new behaviors, providing visual feedback, and providing social interaction.

We tested the initial designs with about 20 people, of varying ages (16–65), both men and women, students, professionals, and general consumers. We found most were quite positive about the Green Machine to be effective in motivating them and changing their behavior in both the short and the long term. A somewhat surprising 35% felt a future view of the world in 100 years was effective even though the news was gloomy based on current trends. We made improvements in icon design, layout, and terminology based on user feedback.

The accompanying two figures show revised screen designs for comparison of energy use and tips for purchasing green products. The first image shows how the user compares energy use with a friend or colleague. Data charts can appear, sometimes with multiple tracks, to show recent time frames, all of which can be customized, for example, a longer term can show performance over a month's time, or longer. The second image shows data about a product purchase that might lead the user to choose one product/company over another because of their "green" attributes. A consumption meter at the top of each screen is a constant reminder of the user's performance. Other screens offer a view into the future 100 years from now to show an estimate of what the earth will be like if people behave as the user now does. Still other screens show social networking and other product evaluation screens to show how a user might use social networks and product/service data to make smarter choices about green behavior.

The Green Machine concept design proved sturdy in tests with potential users. The revised version stands ready for further testing with multicultural users. The mental model and navigation can be built out further to account for shopping, travel, and other energy-consuming activities outside the home. The Green Machine is ready to turn over to companies or governmental sponsors of commercial products and services based on near-term Smart Grid technology developments, including smart-home management and electric/hybrid vehicle management. Even more important, the philosophy, principles, and techniques are readily adapted to other use contexts, namely that of business, both enterprise and small-medium companies, and with contexts beyond ecological data, for example, healthcare. Our company has already developed a follow-on concept design modeled on the Green Machine called the Health Machine.

Coupled with business databases, business use contexts, and business users, the Green Machine for Business might provide another example of how to combine Smart Grid technology with information design and persuasion design for desktop, Web, and mobile applications that can more effectively lead people to changes in business, home, vehicle, and social behavior in conserving energy and using the full potential of the information that the Smart Grid can deliver.

Acknowledgment

This article is based on previous publications (Jean and Marcus, 2009, 2010; Marcus 2010a,b); it includes additional/newer text and newer, revised images.

References

Jean, J., & Marcus, A. (2009). The Green Machine: Going Green at Home. *User Experience (UX)*, *8*(4), 20–22ff.

Marcus, A. (2010a). Green Machine Project. *DesignNet*, *153*(6) June 2010, 114–115 (in Korean).

Marcus, A. (2010b). The Green Machine. *Metering International*, (2), July 2010, South Africa, 90–91.

Marcus, A., & Jean, J. (2010). Going Green at Home: The Green Machine. *Information Design Journal*, *17*(3), 233–243.

8.6 DESIGN FOR EMBODIED INTERACTION

Embodied interaction refers to the ability to involve one's physical body in interaction with technology in a natural way, such as by gestures. Antle (2009) defines embodiment as "how the nature of a living entity's cognition is shaped by the form of its physical manifestation in the world." As she points out, in contrast to the human as information processor view of cognition, humans are primarily active agents, not just "disembodied symbol processors." This means bringing interaction into the human's physical world to involve the human's own physical being in the world.

Embodied interaction, first identified by Malcolm McCullough in *Digital Ground* (2004) and further developed by Paul Dourish in *Where the Action Is* (2001) is central to the idea of phenomenological interaction. Dourish says that embodied interaction is about "how we understand the world, ourselves, and interaction comes from our location in a physical and social world of embodied factors." It has been

described as moving the interaction off the screen and into the real world. Embodied interaction is action situated in the world.

To make it a bit less abstract, think of a person who has just purchased something with "some assembly required." To sit with the instruction manual and just think about it pales in comparison to supplementing that thinking with physical actions in the working environment—holding the pieces and moving them around, trying to fit them this way and that, seeing and feeling the spatial relations and associations among the pieces, seeing the assembly take form, and feeling how each new piece fits.

This is just the reason that physical mockups give such a boost to invention and ideation. The involvement of the physical body, motor movements, visual connections, and potentiation of hand–eye–mind collaboration lead to an embodied cognition far more effective than just sitting and thinking.

Simply stated, embodiment means having a body. So, taken literally, embodied interaction occurs between one's physical body and surrounding technology. But, as Dourish (2001) explains embodiment does not simply refer to physical reality but "the way that physical and social phenomena unfold in real time and real space as a part of the world in which we are situated, right alongside and around us."

As a result, embodiment is not about people or systems per se. As Dourish puts it, "embodiment is not a property of systems, technologies, or artifacts; it is a property of interaction. Cartesian approaches separate mind, body, and thought from action, but embodied interaction emphasizes their duality."

Although tangible interaction (Ishii & Ullmer, 1997) seems to have a following of its own, it is very closely related to embodied interaction. You could say that they are complements to each other. Tangible design is about interactions between human users and physical objects. Industrial designers have been dealing with it for years, designing objects and products to be held, felt, and manipulated by humans. The difference now is that the object involves some kind of computation. Also, there is a strong emphasis on physicality, form, and tactile interaction (Baskinger & Gross, 2010).

More than ever before, tangible and embodied interaction calls for physical prototypes as sketches to inspire the ideation and design process. GUI interfaces emphasized seeing, hearing, and motor skills as separate, single-user, single-computer activities. The phenomenological paradigm emphasizes other senses, action-centered skills, and motor memory. Now we collaborate and communicate and make meaning through physically shared objects in the real world.

In designing for embodied interaction (Tungare et al., 2006), you must think about how to involve hands, eyes, and other physical aspects of the human body

in the interaction. Supplement the pure cognitive actions that designers have considered in the past and take advantage of the user's mind and body as they potentiate each other in problem solving.

Design for embodied interaction by finding ways to shape and augment human cognition with the physical manifestations of motor movements, coupled with visual and other senses. Start by including the environment in the interaction design and understand how it can be structured and physically manipulated to support construction of meaning within interaction.

Embodied interaction takes advantage of several things. One is that it leverages our innate human traits of being able to manipulate with our hands. It also takes advantage of humans' advanced spatial cognition abilities—laying things on the ground and using the relationships of things within the space to support design visually and tangibly.

If we were to try to make a digital version of a game such as Scrabble (example shown later), one way to do it is by creating a desktop application where people operate in their own window to type in letters or words. This makes it an interactive game but not embodied.

Figure 8-14
The Scrabble Flash Cube game.

Another way to make Scrabble digital is the way Hasbro did it in Scrabble Flash Cubes (see later). They made the game pieces into real physical objects with built-in technology. Because you can hold these objects in your hands, it makes them very natural and tangible and contributes to emotional impact because there is something fundamentally natural about that.

Example: Embodied and Tangible Interaction in a Parlor Game

Hasbro Games, Inc. has used embedded technology in producing an electronic version of the old parlor game Scrabble. The simple but fun new Scrabble Flash Cubes game is shown in Figure 8-14. The fact that players hold the cubes, SmartLink letter tiles, in their hands and manipulate and arrange them with their fingers makes this a good example of embodied and tangible interaction.

At the start of a player's turn, the tiles each generate their own letter for the turn. The tiles can read each other's letters as they touch as a player physically shuffles them around. When the string of between two and five letters makes up a word, the tiles light up and beep and the player can try for another word with the same tiles until time is up.

The tiles also work together to time each player's turn, flag duplicates, and display scores. And, of course, it has a built-in dictionary as an authority (however arbitrary it may be) on what comprises a real word.

Ubiquitous Interaction

Ubiquitous interaction is interaction occurring not just on computers and laptops but potentially everywhere in our environment. Interactive devices are being worn by people; embedded within appliances, homes, offices, stereos and entertainment systems, vehicles, and roads; and finding their way into walls, furniture, and objects that we carry.

8.7 UBIQUITOUS AND SITUATED INTERACTION

8.7.1 Ubiquitous, Embedded, and Ambient Computing

The phenomenological paradigm is about ubiquitous computing (Weiser, 1991). Since the term "computing" can conjure a mental image of desktop computers or laptops, perhaps the better term would be ubiquitous interaction with technology, which is more about interaction with ambient computer-like technology worn by people and embedded within appliances, homes, offices, stereos and entertainment systems, vehicles, and roads.

Kuniavsky (2003) concludes that ubiquitous computing requires extra careful attention to design for the user experience. He believes ubiquitous computing devices should be narrow and specifically targeted rather than multipurpose or general-purpose devices looking more like underpowered laptops. And he emphasizes the need to design complete systems and infrastructures instead of just devices.

The concept of embedded computing leans less toward computing in the living environment and more toward computing within objects in the environment. For example, you can attach or embed radio-frequency identification chips and possibly limited GPS capabilities in almost any physical object and connect it wirelessly to the Internet. An object can be queried about what it is and where it is. You can ask your lost possessions where they are (Churchill, 2009).

There are obvious applications to products on store or warehouse shelves and inventory management. More intelligence can be built into the objects, such as household appliances, giving them capabilities beyond self-identification to sensing their own environmental conditions and taking initiative to communicate with humans and with other objects and devices. As example is ambient computing as manifest in the idea of an aware and proactive home.

8.7.2 Situated Awareness and Situated Action

The phenomenological paradigm is also about situated awareness in which the technology and, by the same token, the user are aware of their context. This includes awareness of the presence of others in one's own activity space and their awareness of your virtual presence in their activity spaces. In a social interaction setting, this can help find other people and can help cultivate a feeling of community and belonging (Sellen et al., 2006).

Being situated is all about a sense of "place," the place of interaction within the broader usage context. An example of situated awareness (credit not ours) is a cellphone that "knows" it is in a movie theater or that the owner is in a nonphone conversation; that is, the device or product encompasses knowledge of the rules of human social politeness.