Chapter 6

Scenarios: Translating Goals into Design

In the two previous chapters, we described how to capture qualitative information about users. Through careful analysis of this information and synthesis of user models, we can get a clear picture of our users and their respective goals. We also explained how to prioritize which users are the most appropriate design targets. The missing piece to the puzzle, then, is the process of translating this knowledge into coherent design solutions that meet the needs of users while simultaneously addressing business needs and technical constraints.

This chapter describes a process for bridging the research-design gap. It employs personas as the main characters in set of techniques that rapidly arrive at design solutions in an iterative, repeatable, and testable fashion. This process has three major milestones: defining user requirements; using these requirements to in turn define the fundamental interaction framework for the product; and filling in the framework with ever-increasing amounts of design detail. The glue that holds the processes together is *narrative*: the use of personas to tell stories that point to design.

Narrative as a Design Tool

Narrative, or storytelling, is one of the oldest human activities. Much has been written about the power of narrative to *communicate* ideas. However, narrative can also, through its efficacy at engaging and stimulating creative visualization skills, serve as a powerful tool in generating and validating design ideas (Rheinfrank & Evenson, 1996). Because interaction design is first and foremost the design of behavior that occurs over time, a narrative structure, combined with the support of minimal visualization tools such as the whiteboard, is perfectly suited for envisioning and representing interaction concepts. Detailed refinement calls for more sophisticated visual and interactive tools, but the initial work of defining requirements and frameworks is best done fluidly and flexibly, with minimal reliance on technologies that will inevitably impede ideation.

Scenarios in design

Scenario is a term familiar to usability professionals, commonly used to describe a method of *design problem solving by concretization* (Carroll, 2001): making use of a specific story to both construct and illustrate design solutions. Scenarios are anchored in the concrete, but permit

fluidity; any member of the design team can modify them at will. As Carroll states in his book, *Making Use*:

> Scenarios are paradoxically concrete but rough, tangible but flexible . . . they implicitly encourage 'what-if?' thinking among all parties. They permit the articulation of design possibilities without undermining innovation. . . . Scenarios compel attention to the use that will be made of the design product. They can describe situations at many levels of detail, for many different purposes, helping to coordinate various aspects of the design project.

Carroll's use of **scenario-based design** focuses on describing how *users accomplish tasks* (Carroll, 2001). It consists of an environmental *setting* and includes *agents* or *actors* that are abstracted stand-ins for users, with role-based names such as Accountant or Programmer.

Although Carroll certainly understands the power and importance of scenarios in the design process, the authors see two problems with scenarios as Carroll approaches them:

- ✓ Carroll's scenarios are not concrete enough in their representation of the human actor. It is impossible to design appropriate behaviors for a system without understanding in specific detail the users of the system. Abstracted, role-oriented models are not sufficiently concrete to provide understanding or empathy with users.
- ✓ Carroll's scenarios jump too quickly to the elaboration of tasks without considering the user's goals and motivations that drive and filter these tasks. Although Carroll does briefly discuss goals, he refers only to *goals of the scenario*. These goals are somewhat circularly defined as the completion of specific tasks. Carroll's scenarios begin at the wrong level of detail: User goals need to be considered before user tasks can be identified and prioritized. Without addressing human goals, high-level product definition becomes difficult.

The authors believe that the missing ingredient in scenario-based design methods is the use of personas. A persona provides a sufficiently tangible representation of the user to act as a believable agent in the setting of a scenario. This enhances the designer's ability to empathize with user mental models and perspectives. At the same time, it permits an exploration of how user motivations inflect and prioritize tasks. Because personas model *goals* and not simply tasks, the scope of the problem that scenarios address can also be broadened to include product definition. They help answer the questions, "What should this product *be*?" and "How should this product look and behave?" The authors address the issues surrounding task-based scenarios with the introduction of **persona-based scenarios** — scenarios incorporating the use of personas and goals.

Using personas in scenarios

Persona-based scenarios are concise narrative descriptions of one or more personas using a product to achieve specific goals. Scenarios capture the *non-verbal dialogue* (Buxton, 1990) between artifact and user over time, as well as the structure and behavior of interactive functions. Goals serve as a filter for tasks and as guides for structuring the display of information and controls during the iterative process of constructing the scenarios.

Scenario content and context are derived from information gathered during the Research phase and analyzed during the Modeling phase. Designers role-play personas as the characters in these scenarios (Verplank, et al, 1993), similar to actors performing improvisation. This process leads to real-time synthesis of structure and behavior—typically, at a whiteboard—and later informs the detailed look and feel. Finally, personas and scenarios are used to test the validity of design ideas and assumptions throughout the process. Three types of persona-based scenarios are employed at different points in the process, each time with a successively narrower focus. These scenario types—**context scenarios**, **key path scenarios**, and **validation scenarios**—are described in detail in this chapter.

Persona-based scenarios versus use cases

Scenarios and use cases are both methods of describing a digital system. However, they serve very different functions. Goal-directed scenarios are an iterative means of defining the *behavior* of a product from the standpoint of specific users (personas). This includes not only the functionality of the system, but the priority of functions and the way those functions are expressed in terms of what the user sees and how he interacts with the system.

Use cases, on the other hand, are a technique that has been adopted from software engineering by some usability professionals. They are usually exhaustive descriptions of functional requirements of the system, often of a transactional nature, focusing on low-level user action and system response pairs (Wirfs-Brock, 1993). The precise *behavior* of the system—precisely *how* the system responds—is not, typically, part of a conventional or *concrete* use case; many assumptions about the form and behavior of the system to be designed remain implicit (Constantine and Lockwood, 1999). Use cases permit a complete cataloguing of user tasks for different classes of users, but say little or nothing about how these tasks are presented to the user or how they should be prioritized in the interface. Use cases may be useful in identifying edge cases and for determining that a product is functionally complete, but they should be deployed only in the later stages of design validation.

Envisioning Solutions with Persona-Based Design

As discussed briefly in Chapter 1, the translation from robust models to design solutions really consists of two major phases. **Requirements Definition** answers the broad questions about what a product is and what it should do, and **Framework Definition** answers question about how a product behaves and how it is structured to meet user goals. In this section, we'll discuss both these phases in detail, and the persona-based scenario methodology developed at Cooper by Robert Reimann, Kim Goodwin, Dave Cronin, Wayne Greenwood, and Lane Halley.

Defining the requirements

The Requirements Definition phase determines the *what* of the design: what functions our personas need to use and what kind of information they must access to accomplish their goals. The following five steps comprise this process:

1. Creating problem and vision statements
2. Brainstorming
3. Identifying persona expectations
4. Constructing the context scenario
5. Identifying needs

Although these steps proceed in roughly chronological order, they represent an iterative process. Designers can expect to cycle through steps 3 through 5 several times until the requirements are stable. This is a necessary part of the process and shouldn't be short-circuited. A detailed description of each of these steps follows.

STEP 1: CREATING PROBLEM AND VISION STATEMENTS

Before beginning any process of ideation, it's important for designers to have a clear mandate for moving forward, even if it is a rather high-level mandate. Problem and vision statements provide just such a mandate and are extremely helpful in building consensus among stakeholders before the design process moves forward.

At a high level, the **problem statement** defines the objective of the design (Newman & Lamming, 1995). A design problem statement should concisely reflect a situation that needs changing, for both the personas *and* for the business providing the product to the personas. Often a cause-and-effect relationship exists between business concerns and persona concerns. For example:

> Company X's customer satisfaction ratings are low and market share has diminished by 10% over the past year *because* users don't have adequate tools to perform X, Y, and Z tasks that would help them meet their goal of G.

The connection of business issues to usability issues is critical to drive stakeholders' buy-in to design efforts and to frame the design effort in terms of both user and business goals.

The **vision statement** is an inversion of the problem statement that serves as a high-level design vision or mandate. In the vision statement, you lead with the user's needs, and you transition from those to how business goals are met by the design vision:

> The new design of Product X will help users achieve G by giving them the ability to perform X, Y, and Z with greater [accuracy, efficiency, and so on], and without problems A, B, C that they currently experience. This will dramatically improve Company X's customer satisfaction ratings and lead to increased market share.

The content of both the problem and vision statement should come directly from research and user models. User goals and needs should derive from the primary and secondary personas, and business goals should be extracted from stakeholder interviews.

Problem and vision statements are of most use when you are redesigning an existing product. However, even for new technology products, or products being designed for unexplored market niches, when you formulate user goals and frustrations into problem and vision statements you are helping to establish team consensus on the design activity to follow.

STEP 2: BRAINSTORMING

Brainstorming performed at this early stage of Requirements Definition assumes a somewhat ironic purpose. As designers, you may have been researching and modeling users and the domain for days or even weeks. It is almost impossible that you have not had design ideas percolating in your head. Thus, the reason we brainstorm at this point in the process is to get these ideas out of our heads so we can "let them go," at least for the time being. This serves a primary purpose of eliminating as much designer bias as possible before launching into scenarios, preparing the designers to take on the roles of the primary personas during the scenario process.

Brainstorming should be unconstrained and uncritical—put all the wacky ideas you've been considering (plus some you haven't) out on the table and then be prepared to record them and file them away for safekeeping until much later in the process. It's not likely any of them will be useful in the end, but there might be the germ of something wonderful that will fit into the design framework you later create. Holtzblatt & Beyer (1998) describe a facilitated method for brainstorming that can be useful for getting a brainstorming session started, especially if your team includes non-designers.

Don't spend too much time on the brainstorming step; a few hours (less than half a day) should be more than sufficient for you and your teammates to get all those crazy ideas out of your systems. If you find your ideas are beginning to get repetitious, that's a good time to stop.

STEP 3: IDENTIFYING PERSONA EXPECTATIONS

The expectations that your persona has for a product and its context of use are, collectively, that persona's **mental model** of the product. As we discussed in Chapter 2, it's important that the **represented model** of the interface—how the design behaves and presents itself—should match the user's mental model as closely as possible, rather than reflecting the implementation model of how the product is actually constructed internally.

For each primary persona, you must identify:

- ✓ General expectations and desires each may have about the experience of using the product
- ✓ Behaviors each will expect or desire from the product
- ✓ Attitudes, past experiences, aspirations, and other social, cultural, environmental and cognitive factors that influence these desires

Your persona descriptions may contain enough information to answer some of these questions directly; however, you should return to your research data to analyze the language and grammar of how user subjects define and describe objects and actions that are part of their usage patterns. Some things to look for include:

- ✓ What do the subjects mention first?
- ✓ Which action words (verbs) do they use?
- ✓ Which intermediate steps, tasks, or objects in a process *don't* they mention?

After you have compiled a good list of expectations and influences, do the same for secondary and customer personas and crosscheck similarities and differences.

STEP 4: CONSTRUCTING CONTEXT SCENARIOS

Scenarios are stories about people and their activities (Carroll, 2001). Context scenarios are, in fact, the most story-like of the three types of scenario we employ in that the focus is very much on the persona, her mental models, goals, and activities. Context scenarios describe the broad context in which usage patterns are exhibited and include environmental and organizational (in the case of enterprise systems) considerations (Kuutti, 1995). Context scenarios establish the primary **touch-points** that each primary and secondary persona has with the system (and possibly with other personas via the system) over the course of a day, or some other meaningful length of time that illuminates modes of frequent and regular use. Context scenarios are sometimes, for this reason, called day-in-the-life scenarios.

Context scenarios address questions such as the following (Goodwin, 2002):

- ✓ What is the setting in which the product will be used?
- ✓ Will it be used for extended amounts of time?
- ✓ Is the persona frequently interrupted?
- ✓ Are there multiple users on a single workstation/device?
- ✓ What other products is it used with?
- ✓ How much complexity is permissible, based on persona skill and frequency of use?
- ✓ What primary activities does the persona need to accomplish to meet her goals?
- ✓ What is the expected end result of using the product?

To ensure effective context scenarios, keep them broad and relatively shallow in scope. Resist the urge to dive immediately into interaction detail. It is important to map out the big picture first and systematically identify needs. Doing this and using the steps that follow prevent you from getting lost in design details that may not fit together coherently later.

Context scenarios should *not* represent system behaviors as they currently are. These scenarios represent the brave new world of goal-directed products, so, especially in the initial phases, focus on the goals. Don't yet worry about exactly *how* things will get accomplished—you can initially treat the design as a bit of a magic black box.

Sometimes more than one context scenario is necessary. This is true especially when there are multiple primary personas, but sometimes even a single primary persona may have two or more distinct contexts of use.

Context scenarios are also entirely *textual*. We are not yet discussing form, only the behaviors of the user and the system. This discussion is best accomplished as a textual narrative.

AN EXAMPLE CONTEXT SCENARIO The following is an example of a first iteration of a context scenario for a primary persona for a PDA/phone convergence device and service: Vivien Strong, a real-estate agent in Indianapolis. Vivien's goals are to balance work and home life, cinch the deal, and make each client feel like he is her *only* client.

Vivien's context scenario might be as follows:

1. Getting ready in the morning, Vivien uses her phone to check e-mail. It has a large enough screen and quick connection time so that it's more convenient than booting up a computer as she rushes to make her daughter, Alice, a sandwich for school.
2. Vivien sees an e-mail from her newest client, Frank, who wants to see a house this afternoon. Vivien entered his contact info a few days ago, so now she can call him with a simple action right from the e-mail screen.
3. While on the phone with Frank, Vivien switches to speakerphone so she can look at the screen while talking. She looks at her appointments to see when she's free. When she creates a new appointment, the phone automatically makes it an appointment with Frank, because it knows with whom she is talking. She quickly keys the address of the property into the appointment as she finishes her conversation.
4. After sending Alice off to school, Vivien heads into the real-estate office to gather the papers she needs for the plumber working on another property. Her phone has already updated her Outlook appointments, so the rest of the office knows where she'll be in the afternoon.
5. The day goes by quickly, and she's running a bit late. As she heads towards the property she'll be showing Frank, the phone alerts her that her appointment is in 15 minutes. When she flips open the phone, it shows not only the appointment, but a list of all documents related to Frank, including e-mails, memos, phone messages, call logs to Frank's number, and even thumbnail pictures of the property that Vivien sent as e-mail attachments. Vivien presses the call button, and the phone automatically connects to Frank because it knows her appointment with him is soon. She lets him know she'll be there in 20 minutes.
6. Vivien knows the address of the property, but is a bit unsure exactly where it is. She pulls over and taps the address she put into the appointment. The phone downloads directions along with a thumbnail map showing her location relative to the destination.
7. Vivien gets to the property on time and starts showing it to Frank. She hears the phone ring from her purse. Normally while she is in an appointment, the phone will automatically transfer directly to voicemail, but Alice has a code she can press to get through. The phone knows it's Alice calling, and uses a distinctive ring tone.

8. Vivien takes the call—Alice missed the bus and needs a pickup. Vivien calls her husband to see if he can do it. She gets his voicemail; he must be out of service range. She tells him she's with a client, and asks if he can get Alice. Five minutes later the phone makes a brief tone Vivien recognizes as her husband's; she sees he's sent her an instant message: "I'll get Alice; good luck on the deal!"

Note how the scenario remains at a fairly high level, not getting too specific about interfaces or technologies. It's important to create scenarios that are within the realm of technical possibility, but at this stage the details of reality aren't yet important. It's always possible to scale back; we are ultimately trying to describe an *optimal*, yet still feasible experience. Note also how the activities in the scenario tie back to Vivien's goals and try to strip out as many tasks as possible.

PRETENDING IT'S MAGIC A powerful tool in the early stages of developing scenarios is to *pretend the interface is magic* (Cooper, 1999). If your persona has goals and the product has magical powers to meet them, how simple could the interaction be? This kind of thinking is useful to help designers look outside the box. Magical solutions obviously won't suffice, but figuring out creative ways to technically accomplish interactions that are as close to magical solutions as possible(from the personas' perspective) is the essence of great interaction design. Products that meet goals with the minimum of hassle and intrusion seem almost magical to users. Some of the interactions in the preceding scenario may seem a bit magical, but all are possible with technology available today. It's the goal-directed behavior, not the technology alone, that provides the magic.

In early stage design, pretend the interface is magic.

STEP 5: IDENTIFYING NEEDS

After you are satisfied with an initial draft of your context scenario, you can begin to analyze it to extract the personas' needs. These needs consist of *objects* and *actions* (Shneiderman, 1998) as well as *contexts*. The authors prefer not to think of needs as identical to tasks. The implication is that tasks must be manually performed by the user, whereas the term needs implies simply that certain objects need to exist and that certain actions on them need to happen (whether initiated by the user or the system) in certain contexts. Thus, a need from the scenario above might be:

Call (action) a person (object) directly from an appointment (context)

If you are comfortable extracting needs in this format, it works quite well; otherwise, you can separate them as described in the following sections.

DATA NEEDS Personas' data needs are the objects and information that must be represented in the system. Charts, graphs, status markers, document types, attributes to be sorted, filtered, or manipulated, and graphical object types to be directly manipulated (in authoring and art software) are all examples of data needs.

FUNCTIONAL NEEDS Functional needs are the operations that need to be performed on the objects of the system and which are eventually translated into interface controls. Functional needs also define places or containers where objects or information in the interface must be displayed.

CONTEXTUAL NEEDS AND REQUIREMENTS Contextual needs describe relationships between sets of objects or sets of controls, as well as possible relationships between objects and controls. This can include which types of objects to display together to make sense for workflow or to meet specific persona goals, as well as how certain objects must interact with other objects (for example, when choosing items for purchase, a summed list of items already selected needs to be visible). Other contextual requirements include considerations of the product's physical environment(s) (an office, on the go, indoors, outdoors.) and the skills and capabilities of the personas using the product.

OTHER REQUIREMENTS After you've gone through the exercise of pretending it's magic, it's important to get a firm idea of the realistic requirements of the business and technology you are designing for (although we hope that designers have some influence over technology choices when it directly affects user goals).

- ✓ **Business requirements** can include development timelines, regulations, pricing structures, and business models.
- ✓ **Technical requirements** can include weight, size, form-factor, display, power constraints, and software platform choices.
- ✓ **Customer and partner requirements** can include ease of installation, maintenance, configuration, support costs, and licensing agreements.

Now your design team should have a mandate in the form of the problem and vision statements, a rough, creative overview of how the product is going to address user goals in the form of context scenarios, and a reductive list of needs and requirements extracted from your research, user models, and the scenarios. Now you are ready to delve deeper into the details of your product's behaviors, and begin to consider how the product and its functions will be represented. You are ready to define the framework of the interaction.

Defining the interaction framework

The Requirements Definition phase sets the stage for the core of the design effort: defining the interaction framework of the product. The interaction framework defines not only the skeleton of the interaction — its structure — but also the flow and behavior of the product. The following six steps describe the process of defining the interaction framework:

1. Defining form factor and input methods
2. Defining views
3. Defining functional and data elements
4. Determining functional groups and hierarchy

5. Sketching the interaction framework
6. Constructing key path scenarios

Like previous processes, this is not a linear effort, but requires iteration. The steps are described in more detail in the following sections.

STEP 1: DEFINING FORM FACTOR AND INPUT METHODS

The first step in creating a framework is defining the form factor of the product you'll be designing. Is it a Web application that will be viewed on a high-resolution computer screen? Is it a phone that must be small, light, low-resolution, and visible in the dark and as well as in bright sunlight? Is it a kiosk that must be rugged to withstand a public environment with thousands of distracted, novice users? What are the constraints that each of these imply for any design? Answering these questions sets the stage for all subsequent design efforts.

After you have defined this basic *posture* (see Chapter 8) of the product, you should then determine the valid input methods for the system: Keyboard, mouse, keypad, thumboard, touch screen, voice, game controller, remote control, and many other possibilities exist. Which combination is appropriate for your primary and secondary personas? What is the *primary* input method for the product?

STEP 2: DEFINING VIEWS

The next step, after basic form factor and input methods are defined, is to consider which primary screens or states the product can be in. Initial context scenarios give you a feel for what these might be: They may change or rearrange somewhat as the design evolves (particularly in step 4), but it is often helpful to put an initial stake in the ground to serve as a means for organizing your thoughts. If you know that a user has several end goals and needs that don't closely relate to each other in terms of data overlap, it might be reasonable to define separate views to address them. On the other hand, if you see a cluster of related needs (for example, to make an appointment, you need to see a calendar and possibly contacts), you might consider defining a view that incorporates all these together, assuming the form factor allows it.

STEP 3: DEFINING FUNCTIONAL AND DATA ELEMENTS

Functional and data elements are the visible representations of functions and data in the interface. They are the concrete manifestations of the functional and data needs identified during the Requirements Definition phase. Where those needs were purposely described in terms of real-world objects and actions, functional and data elements are described in the language of user interface representations:

- ✓ Panes, frames, and other containers on screen
- ✓ Groupings of on-screen and physical controls
- ✓ Individual on-screen controls
- ✓ Individual buttons, knobs, and other physical affordances on a device
- ✓ Data objects (icons, listed items, images, graphs) and associated attributes

In early framework iterations, containers are the most important to specify; later as you focus on the design of individual containers, you will get to more detailed interface elements.

Many persona needs will spawn multiple interface elements to meet those needs. For example, Vivien needs to be able to telephone her contacts. Functional elements to meet that need include:

- ✓ Voice activation (voice data associated with contact)
- ✓ Assignable quick-dial buttons
- ✓ Selecting from a list of contacts
- ✓ Selecting the name from e-mail header, appointment, or memo
- ✓ Auto-assignment of a call button in proper context (appointment coming up)

Multiple vectors are often a good idea, but sometimes not all possible vectors will be useful to the persona. Use persona goals, design principles, and patterns (see Chapter 7), as well as business and technical constraints to winnow your list of elements for meeting particular needs. You will also need to determine data elements. Some of Vivien's data elements might include appointments, memos, to-do items, and messages.

STEP 4: DETERMINING FUNCTIONAL GROUPS AND HIERARCHY

After you have a good list of top-level functional and data elements, you can begin to group them into functional units and determine their hierarchy (Shneiderman, 1998). Because these elements facilitate specific tasks, the idea is to group elements to best facilitate the persona's flow (see Chapter 9) both within a task and between related tasks. Some issues to consider include:

- ✓ Which elements need a large amount of real estate and which do not?
- ✓ Which elements are *containers* for other elements?
- ✓ How should containers be arranged to optimize flow?
- ✓ Which elements are used together and which aren't?
- ✓ In what sequence will a set of related elements be used?
- ✓ What interaction patterns and principles apply?
- ✓ How do the personas' mental models affect organization? (Goodwin, 2002)

The most important initial step is determining the top-level container elements for the interface, and how they are best arranged given the form factor and input methods that the product requires. Containers for objects that must be compared or used together should be adjacent to each other. Objects representing steps in a process should, in general, be adjacent and ordered sequentially. Use of interaction design principles and patterns is extremely helpful at this juncture; Chapter 7 and Part II of this book provide many principles that can be of assistance at this stage of organization.

STEP 5: SKETCHING THE INTERACTION FRAMEWORK

You may want to sketch different ways of fitting top-level containers together in the interface. Sketching the framework is an iterative process that is best performed with a small, collaborative group of one or two interaction designers and a visual or industrial designer. This visualization of the interface should be extremely simple at first: boxes representing each functional group and/or container with names and descriptions of the relationships between the different areas (see Figure 6-1).

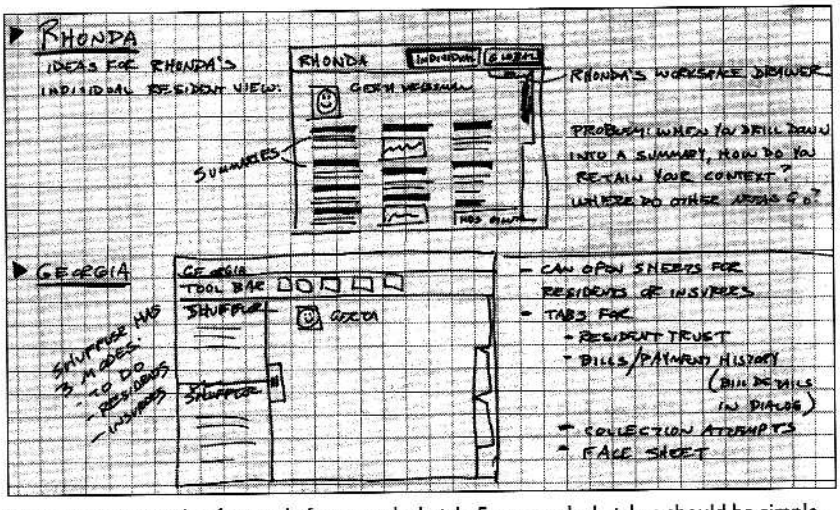

Figure 6-1: Example of an early framework sketch. Framework sketches should be simple, starting with rectangles, names, and simple descriptions of relationships between functional areas. Details can be visually hinted at to give an idea of contents, but don't fall into the trap of designing detail at this stage.

Be sure to look at the entire, top-level framework first; don't let yourself get distracted by the details of a particular area of the interface. There will be plenty of time to explore the design at the widget level and, by going there too soon, you risk a lack of coherence in the design later.

STEP 6: CONSTRUCTING KEY PATH SCENARIOS

Key path scenarios result from exploring details hinted at, but not addressed, in the context scenarios. Key path scenarios describe at the task level the primary actions and pathways through the interface that the persona takes with the greatest frequency, often on a daily basis. In an e-mail application, for example, viewing and composing mail are key path activities; configuring a new mail server is not.

Key path scenarios generally require the greatest interaction support. New users must master key path interactions and functions quickly, so they need to be supported by built-in pedagogy (see Chapters 18 and 27). However, because these functions are used frequently, users do not remain dependent on that pedagogy for long: They will rapidly demand shortcuts. In addition, as users become very experienced, they will want to customize daily use interactions so that they conform to their individual work styles and preferences.

SCENARIOS AND STORYBOARDING Unlike the goal-oriented context scenarios, key path scenarios are more task-oriented; focusing on task details broadly described and hinted at in the context scenarios (Kuutti, 1995). This doesn't mean that goals are ignored — goals and persona needs are the constant measuring stick throughout the design process, used to trim unnecessary tasks and streamline necessary ones. However, **key path scenarios** *must describe in exacting detail the precise behavior of each major interaction* and provide a walkthrough (Newman & Lamming, 1995) of each major pathway.

Typically, key path scenarios begin at a whiteboard and reach a reasonable level of detail. At some point, depending on the complexity and density of the interface, it becomes useful to graduate to computer-based tools. The authors are fond of Microsoft PowerPoint as a tool for aiding in the **storyboarding** of key path scenarios. Storyboarding is a technique borrowed from filmmaking and cartooning. Each step in an interaction, whether between the user and the system, multiple users, or some combination thereof (Holtzblatt & Beyer, 1998) can be portrayed on a slide, and clicking through them provides a reality check for the coherence of the interaction (see Figure 6-2). PowerPoint is sufficiently fast and low-resolution to allow rapid drawing and iterating without succumbing to creating excessive detail.

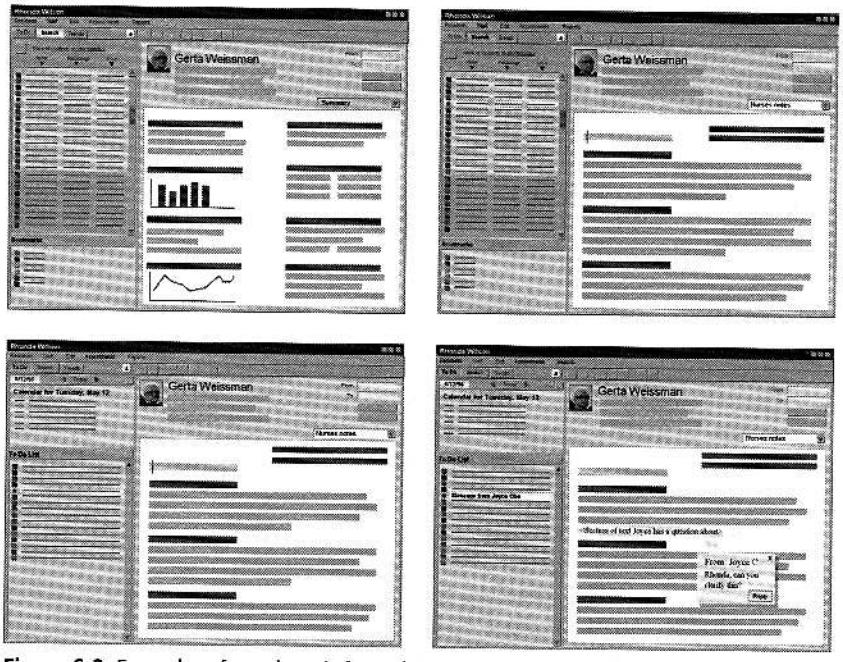

Figure 6-2: Examples of storyboards from Shared Healthcare Systems' Orcas product.

PRETENDING THE SYSTEM IS HUMAN Just as pretending it's magic is a powerful tool for constructing concept-level, context scenarios, pretending the system is human is a powerful tool at the interaction-level appropriate to key path scenarios. The principle is simple (and discussed at length in Chapter 14): Interactions with a digital system should be similar in tone and helpfulness to interactions with a polite, considerate human (Cooper, 1999). As you construct your interactions,

you should ask yourself: Is the primary persona being treated humanely by the product? What would a thoughtful, considerate interaction look like? In what ways can the software offer helpful information without getting in the way? How can it minimize the persona's effort in reaching his goals? What would a helpful human do?

PRINCIPLES AND PATTERNS Critical to the translation of key path scenarios to storyboards (as well as the grouping of elements in step 3) is the application of general interaction principles and specific interaction patterns. These tools leverage years of interaction design knowledge—not to take advantage of such knowledge would be tantamount to re-inventing the wheel. Key path scenarios provide an inherently top-down approach to interaction design, iterating through successively more-detailed design structures from main screens down to tiny subpanes or dialogs. Principles and patterns add a bottom-up approach to balance the process. Principles and patterns can be used to organize elements at all levels of the design. Chapter 7 discusses the uses and types of principles and patterns in detail, and the chapters of Section Three provide a wealth of useful interaction principles appropriate to this step in the process.

Refining the form and behavior

When a solid, stable framework definition is reached, designers see the remaining pieces of the design begin to smoothly fall into place: Every iteration of the key path scenarios adds detail that strengthens the overall coherence and flow of the product. At this stage, a smooth transition can be made into the Refinement phase, where the final translation of the design into a more concrete form occurs. The process of refinement includes these steps:

1. Drafting the look and feel
2. Constructing validation scenarios
3. Finalizing the design

In this phase, principles and patterns remain important in giving the design a fine formal and behavioral finish. Chapter 19 and many chapters in Section Three provide useful principles for the Refinement phase. It is also critical for the development team to be involved from the start of the Refinement phase—now that the design has a solid conceptual and interaction basis, developer input is critical to creating a finished design that can both be built and remain true to concept.

STEP 1: DRAFTING THE LOOK AND FEEL

By the time the interaction framework has reached the key path scenario stage, interaction designers should begin to work with visual interface designers to develop both the branding and functional components of the visual design (see Chapter 19 for some basic visual design principles). Similarly, industrial designers should be involved at this stage. Visual and industrial designers should walk through the key path scenarios with interaction designers. The different design perspectives are complementary, and result in a better, more desirable product as long as all parties are able to keep within the character of the persona for whom the scenario is being created.

Visual and industrial designers (or interaction designers with these skills) should begin to translate detailed wireframes and storyboards to representative, full-resolution bitmap screens (see Figure 6-3).

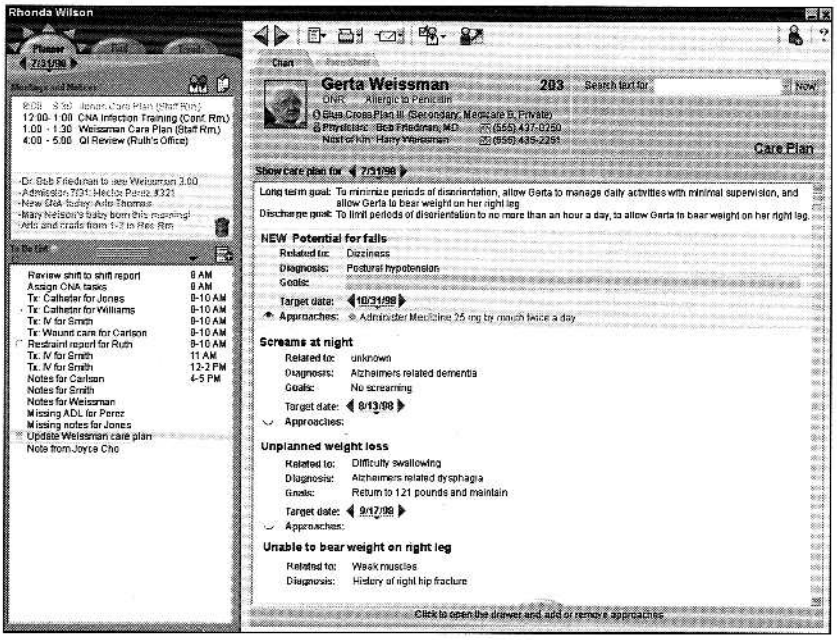

Figure 6-3: Full-resolution bitmap screens for SHS Orcas based on the storyboards from the previous figure. Note that there are minor changes to the layout that naturally result from the realities of pixels and screen resolution. Visual and interaction designers need to work closely together at this stage to ensure that visual changes to the design continue to reinforce appropriate product behaviors and meet the goals of the primary personas.

Every primary view and dialog should be addressed. Visual designers should create style guides so that developers can apply visual design elements consistently to the low-level, lower priority parts of the interface that the designers may have lacked time and resources to complete themselves.

STEP 2: CONSTRUCTING VALIDATION SCENARIOS

If a user performs a task frequently, its interaction must be well crafted. Likewise, if a task is critical but performed infrequently, its interaction, although designed with different objectives, must still be well designed. After you have walked through key path scenarios extensively and storyboarded, it is time to shift focus to less frequently used corners of the interface. You develop these scenarios in the same manner as key path scenarios and prioritize them in this general order:

KEY PATH VARIANTS Key path variant scenarios are less-traveled interactions that split off from key pathways at some point along the persona's decision tree. These could include lesser-used functions that still belong on a main palette or toolbar, less frequently used views of documents, and major dialogs that are not critical to the basic operation of the product. Key path variants have all the same requirements as key path scenarios, with a slightly greater emphasis on pedagogy because they are used less often.

NECESSARY USE Necessary use scenarios include all actions that *must* be performed, but that are performed infrequently. Purging databases, configuring, and making other exceptional requests might fall into this category. Necessary use interactions demand pedagogy because they are used infrequently: Users may forget how to access the function or how to perform tasks related to it. However, users won't ever require parallel interaction idioms such as keyboard equivalents because the function is rarely used. Also, because of their infrequent use, necessary functions don't need to be user-customizable.

EDGE CASE USE Edge case use scenarios, as the name implies, are activities that are both optional and infrequent. Programmers often want to emphasize the edge cases handled by validation scenarios because of their natural tendency to view all features and functions as equally important, but edge cases should not be the focus of the design effort. Designers can't ignore edge case functions and situations, but the interaction needed for them is of lower priority and can, typically, be buried fairly deeply in the interface. Although the *code* may succeed or fail on its capability to successfully handle edge cases, the *product* will succeed or fail on its capability to successfully handle daily use and necessary cases.

FINALIZE THE DESIGN

After you have checked the design through validation scenarios, you are ready, in concert with your visual and industrial design teams, to craft the final look and feel of the design. From there, you can produce a variety of outputs, including a printed form and behavior specification. The form and behavior spec includes screen mockups and storyboards with callouts sufficiently detailed for a programmer to code from. You can, instead, produce an interactive prototype in HTML or an application like Flash or Director that serves much the same purpose. Regardless of your choice of design deliverable, your team should continue to work closely with the development team throughout implementation to ensure that the design vision is accurately translated from the design document to a final product that is faithful to it.