6

The Foundations of Design: Scenarios and Requirements

In the two previous chapters, we talked about how to gather qualitative information about users and create models using that information. Through careful analysis of user research and synthesis of personas and other user models, we create a clear picture of our users and their respective goals. This brings us, then, to the crux of the whole method: how we use this understanding of people to create design solutions that satisfy and inspire users, while simultaneously addressing business goals and technical constraints.

This chapter describes the first part of a process for bridging the research-design gap. It employs personas as the main characters in a set of techniques that rapidly arrive at design solutions in an iterative, repeatable, and testable fashion. This process has four major activities: developing stories or *scenarios* as a means of imagining ideal user interactions, using those scenarios to define *requirements*, using these requirements in turn to 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*: using personas to create stories that point to design.

Scenarios: 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 is also one of our most powerful creative methods. From a very young age, we are accustomed to using stories to think about possibilities, and this is an incredibly effective way to *imagine* a new and better future for our users. Imagining a story about a person using our product leverages our creativity to a greater power than when we just imagine a better form factor or configuration of screen elements. Further, because of the intrinsically social aspect of narrative, it is a very effective and compelling way to share good ideas among team members and stakeholders. Ultimately, experiences designed around narrative tend to be more comprehensible and engaging for users because they are structured around a story.

Evidence of the effectiveness of narrative as a design tool is all around us. The famous Disney Imagineers would be lost without the modern-day myths they use as the foundation for the experiences they build. Much has been written about this idea: Brenda Laurel explored the concept of structuring interaction around dramatic principles in her 1991 book *Computers as Theater*, where she urges us to ". . . focus on designing the action. The design of objects, environments, and characters is all subsidiary to this central goal."[1] John Rheinfrank and Shelley Evenson also talk about the power of "stories of the future" for developing conceptually complex interactive systems,[2] and John Carroll has created a substantial body of work about scenario-based design, which we discuss later in this chapter.

Narrative also lends itself to effective visual depictions of interactive products. Because interaction design is first and foremost the design of behavior that occurs over time, a narrative structure, combined with the support of fast and flexible visualization tools (such as the humble whiteboard), is perfectly suited for motivating, envisioning, representing, and validating interaction concepts.

Interaction design narratives are quite similar to the comic-book-like sequences called storyboards that are used in the motion picture industry. They share two significant characteristics: plot and brevity. Just as storyboards breathe life into a movie script, design solutions should be created and rendered to follow a plot — a story. Putting too much detail into the storyboards simply wastes time and money and has a tendency to tie us to suboptimal ideas simply because drawing them consumes significant resources.

In the initial requirements definition phase we are free to focus only on the "plot points," allowing us to be fluid as we explore design concepts. Because they are enough to convey the action and the potential experience, many millions of Hollywood dollars

have been invested on the basis of simple pencil sketches or line drawings. By focusing on the narrative, we are able to quickly and flexibly arrive at a high-level design solution without getting bogged-down by the inertia and expense inherent to high-production-value renderings (though such renderings are certainly appropriate once a working design framework is in place).

Scenarios in design

In the 1990s, substantial work was done by the HCI (Human-Computer Interaction) community around the idea of use-oriented software design. From this work came the concept of the **scenario**, commonly used to describe a method of *design problem solving by concretization*: making use of a specific story to both construct and illustrate design solutions. These concepts are discussed by John Carroll, 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.*[3]

Carroll's use of **scenario-based design** focuses on describing how *users accomplish tasks*. 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, we've found two shortcomings with scenarios as Carroll approaches them:

- Carroll's concept of the actor as an abstracted, role-oriented model is not sufficiently concrete to provide understanding of or empathy with users. It is impossible to design appropriate behaviors for a system without understanding the users of the system in specific detail.
- 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 circularly defined as the completion of specific tasks. In our experience, user goals must be considered before user tasks can be identified and prioritized. Without addressing the motivation of human behavior, high-level product definition can be difficult and misguided.

The missing ingredient in Carroll's scenario-based design methods is the use of personas. A persona provides a tangible representation of the user to act as a believable agent in the setting of a scenario. In addition to reflecting current behavior patterns and motivations, personas enable the exploration of how user motivations should inflect and prioritize tasks in the future. Because personas model *goals* and not simply tasks, the scope of the problems addressed by scenarios can be broadened to include those related to product definition. They help answer the questions, "What should this product *do*?" and "How should this product look and behave?"

Using personas in scenarios

Persona-based scenarios are concise narrative descriptions of one or more personas using a product to achieve specific goals. They allow us to start our designs from a story describing an ideal experience from the persona's perspective, focusing on people, and how they think and behave, rather than on technology or business goals.

Scenarios can capture the *nonverbal dialogue*[4] between the user and a product, environment, or system 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,[5] 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.

Different types of scenarios

The Goal-Directed Design method employs three types of persona-based scenarios at different points in the process, each with a successively more interface-specific focus. The first — the **context scenario** — is used to explore, at a high level, how the product can best serve the needs of the personas. (We used to call these "day-in-the-life scenarios," but found that term excessively broad.) The context scenarios are created before any design is performed and are written from the perspective of the persona, focused on human activities, perceptions, and desires. It is in the development of this kind of scenario that the designer has the most leverage to imagine an

ideal user experience. More detail about the creation of this type of scenario can be found later in this chapter, under Step 4 in the Requirements Definition process.

Once the design team has defined the product's functional and data elements, and developed a Design Framework (as described in Chapter 7), a context scenario is revised to become a **key path scenario** by more specifically describing user interactions with the product and by introducing the vocabulary of the design. These scenarios focus on the most significant user interactions, always maintaining attention on how a persona uses the product to achieve their goals. Key path scenarios are iteratively refined along with the design as more and more detail is developed.

Throughout this process, the design team uses **validation scenarios** to test the design solution in a variety of situations. These scenarios tend to be less detailed and typically take the form of a number of "what if . . ." questions about the proposed solutions. More detail about development and use of key path and validation scenarios can be found in Chapter 7.

Persona-based scenarios versus use cases

Scenarios and use cases are both methods of describing a user's interaction with a 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 she interacts with the system.

Use cases, on the other hand, are a technique based on exhaustive descriptions of functional requirements of the system, often of a transactional nature, focusing on low-level user action and accompanying system response.[6] 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.[7] 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. In our experience, the biggest shortcoming of traditional use cases as a basis for interaction design is their tendency to treat all possible user interactions as equally likely and important. This is indicative of their origin in software engineering rather than interaction design. They 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.

Requirements: The "What" of Interaction Design

The Requirements Definition phase determines the *what* of the design: what information and capabilities our personas require to accomplish their goals. It is absolutely critical to define and agree upon the *what* before we move on to the next question: *how* the product looks, behaves, operates, and feels. Conflating these two questions can be one of the biggest pitfalls in the design of an interactive product. Many designers are tempted to jump right into active design and render possible solutions. Regardless of how creative and skillful you are, we urge you not to do this. It runs the risk of turning into a never-ending circle of iteration; proposing a solution without clearly defining and agreeing upon the problem leaves you without a clear method of evaluating the fitness of the design. In lieu of such a method, you, your stakeholders, and your clients are likely to resort to taste and gut instinct, which have a notoriously low success ratio with something as complex as an interactive product.

Define *what* the product will do before you design *how* the product will do it.

It's important to note that our concept of a "requirement" here is much different from the way the term is commonly misused in the industry. In many product-development organizations, "requirement" has come to be synonymous with "feature" or "function." While there is clearly a relationship between requirements and functions (which we leverage as a key part of our design process, as you will see in the next chapter), we suggest that you think of requirements as synonymous with *needs*. Put another way, at this point, you want to rigorously define the human and business needs that your product must satisfy.

Another critical reason not to conflate requirements with features is that in figuring out the best way to meet a particular human need, an interaction designer has an extraordinary amount of leverage to create a powerful and compelling product. Think, for example, about designing a data analytics tool to help an executive better understand the state of his business. If you jump right to the *how* without understanding the *what*, you might assume that the output of the tool should be reports. It would be easy to come to this conclusion; if you went out and performed user research, you probably would have noticed that reports are a very widespread

and accepted solution. However, if you imagine some scenarios and analyze your users' actual requirements, you might realize that your executive actually needs a way to recognize exceptional situations before opportunities are missed or problems arise, as well a way to understand emerging trends in the data. From here, it isn't difficult to see that static, flat reports are hardly the best way to meet these needs. With such a solution, your executive has to do the hard work of scrutinizing several of these reports to find the significant data underlying such exceptions and trends. Much better solutions might include data-driven exception reporting or real-time trend monitors.

A final reason to separate problem and solution is that such an approach provides the maximum flexibility in the changing face of technological constraints and opportunities. By clearly defining the user need, designers can then work with technologists to find the best solutions, without compromising the product's ability to help people achieve their goals. Working in this manner, the product definition is not at risk when the implementation runs into problems, and it becomes possible to plan long-term technology development so that it can provide increasingly sophisticated ways of meeting user needs.

As we've mentioned briefly, these requirements come from several sources. Personas' previous experiences and mental models often result in some baseline expectations of the product. We derive the bulk of the user requirements from analyzing ideal usage scenarios, and understand business and technical requirements from our stakeholder interviews. Our Goal-Directed process for defining product requirements is described below.

Requirements Definition Using Personas and Scenarios

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 questions about how a product behaves and how it is structured to meet user goals. In this section, we'll discuss Requirements Definition in detail, followed by a discussion of the Framework Definition in Chapter 7. The methods described are based upon the persona-based scenario methodology developed at Cooper by Robert Reimann, Kim Goodwin, Dave Cronin, Wayne Greenwood, and Lane Halley.

The Requirements Definition process comprises the following five steps (which are described in detail in the remainder of this chapter):

1. Creating problem and vision statements
2. Brainstorming
3. Identifying persona expectations
4. Constructing context scenarios
5. Identifying requirements

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 the process of ideation, it's important for designers to have a clear mandate for moving forward. While the Goal-Directed method aims to comprehensively define the product through personas, scenarios, and requirements, it is often useful at this point to define what direction these scenarios and requirements should be headed in. At this point in the process, we already have a sense of which users we're targeting and what their goals are, but lacking a clear product mandate, there is still room for considerable confusion. 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 purpose of the design initiative.[8] 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 objective 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 statements 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 useful both when you are redesigning an existing product and for new technology products or products being designed for unexplored market niches, when formulating user goals and frustrations into problem and vision statements helps to establish team consensus and attention on the priorities of design activity to follow.

Step 2: Brainstorming

At the early stage of Requirements Definition, brainstorming assumes a somewhat ironic purpose. At this point in the project, we have been researching and modeling users and the domain for days or even months, and it is almost impossible to avoid having developed some preconceptions about what the solution looks like. However, we'd ideally like to create context scenarios without these prejudgments, and instead really focus on how our personas would likely want to engage with the product. The reason we brainstorm at this point in the process is to get these ideas out of our heads so that we can record them and thereby "let them go" for the time being.

The primary purpose here is to eliminate as much preconception as possible, allowing designers to be open-minded and flexible as they use their imagination to construct scenarios, and use their analytic minds to derive requirements from these scenarios. A side benefit of brainstorming at this point in the process is to switch your brain into "solution mode." Much of the work performed in the Research and Modeling phases is analytical in nature, and it takes a different mindset to come up with inventive designs.

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 necessarily 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. Karen Holtzblatt and Hugh Beyer describe a facilitated method for brainstorming that can be useful for getting a brainstorming session started, especially if your team includes nondesigners.[9]

Don't spend too much time on the brainstorming step; a few hours 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 getting repetitious, or the popcorn stops popping, that's a good time to stop.

Step 3: Identifying persona expectations

As we discussed in Chapter 2, a person's **mental model** is their own internal representation of reality — the way they think about or explain something to themselves. Mental models are deeply ingrained and are often the result of a lifetime of experience. People's expectations about a product and the way it works are highly informed by their mental model.

Returning to our discussion in Chapter 2, it's absolutely critical 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.

In order to accomplish this, we must formally record these expectations. They will be an important source of requirements. For each primary and secondary persona, you must identify:

- Attitudes, experiences, aspirations, and other social, cultural, environmental, and cognitive factors that influence the persona's expectations
- General expectations and desires the persona may have about the experience of using the product
- Behaviors the persona will expect or desire from the product
- How that persona thinks about basic elements or units of data (for example, in an e-mail application, the basic elements of data might be messages and people)

Your persona descriptions may contain enough information to answer these questions directly; however, your research data will remain a rich resource. Use it to

analyze how interview subjects define and describe objects and actions that are part of their usage patterns, along with the language and grammar they use. 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? (Hint: These might not be terribly important to the way they think about things.)

Step 4: Constructing context scenarios

While all scenarios are stories about people and their activities, context scenarios are the most storylike of the three types we employ. The focus is on the persona's activities, as well as her motivations and mental model. **Context scenarios** describe the broad context in which usage patterns are exhibited and include environmental and organizational (in the case of enterprise systems) considerations. [10]

As we discussed above, *this is where design begins.* As you develop context scenarios, you should be focusing on how the product you are designing can best help your personas achieve their goals. Context scenarios establish the primary touch points that each primary and secondary persona has with the system (and possibly with other personas) over the course of a day or some other meaningful length of time.

Context scenarios should be broad and relatively shallow in scope. They should not describe product or interaction detail but rather should focus on high-level actions from the user's perspective. It is important to map out the big picture first so that we can systematically identify user requirements. Only then will we be able to design appropriate interactions and interfaces.

Context scenarios address questions such as the following:

- In what setting(s) will the product be used?
- Will it be used for extended amounts of time?
- Is the persona frequently interrupted?
- Are there multiple users on a single workstation or device?
- With what other products will it be used?
- What primary activities does the persona need to perform to meet her goals?
- What is the expected end result of using the product?
- How much complexity is permissible, based on persona skill and frequency of use?

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 should initially treat the design as a bit of a magic black box.

In most cases, 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 personal digital assistant (PDA) type phone, including both the device and its service. Our persona is Vivien Strong, a real-estate agent in Indianapolis, whose goals are to balance work and home life, close the deal, and make each client feel like he is her *only* client.

Vivien's context scenario:

1. While getting ready in the morning, Vivien uses her phone to check her 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. The device has his contact info, so now she can call him with a simple action right from the e-mail.
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 enters 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 some papers for another appointment. 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, and call logs to Frank's number. 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!"

Notice how the scenario remains at a fairly high level, without 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. We want to leave the door open for truly novel solutions, and it's always possible to scale back; we are ultimately trying to describe an *optimal*, yet still feasible, experience. Also notice 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*. 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 stages of design, pretend the interface is magic.

Step 5: Identifying requirements

After you are satisfied with an initial draft of your context scenario, you can analyze it to extract the personas' needs or requirements. These **requirements** can be thought of as consisting of *objects*, *actions*, and *contexts*.[11] And remember, as we discuss above, we prefer not to think of requirements as identical to features or tasks. 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 may find it helpful to separate them into data, functional, and contextual requirements, as described in the following sections.

Data requirements

Personas' data needs are the objects and information that must be represented in the system. Using the semantics described above, it is often useful to think of data requirements as the objects and adjectives related to those objects. Common examples include accounts, people, documents, messages, songs, images, as well as attributes of those such as status, dates, size, creator, subject, and so on.

Functional requirements

Functional needs are the operations or actions that need to be performed on the objects of the system and which are typically translated into interface controls. These can be thought of as the *actions* of the product. Functional needs also define places or containers where objects or information in the interface must be displayed. (These are clearly not actions in and of themselves but are usually suggested by actions.)

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.

- **Brand and experience requirements** reflect attributes of the experience you would like users and customers to associate with your product, company, or organization.
- **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.

Having performed these steps, you should now have 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.

Notes

1. Laurel, *Computers as Theater,* 134
2. Rheinfrank and Evenson, 1996
3. Carroll, 2001
4. Buxton, 1990
5. Verplank, et al, 1993
6. Wirfs-Brock, 1993
7. Constantine and Lockwood, 1999
8. Newman and Lamming, 1995
9. Holtzblatt and Beyer, 1998
10. Kuutti, 1995
11. Shneiderman, 1998