Since stakeholders and users generally aren't available during this stage, you also serve as the ambassador of their interests. If it becomes necessary to consult stakeholders on certain key questions, you'll need to ensure this is done in the best way possible. Every contact with stakeholders needs to be viewed as an opportunity to improve their buy-in and maintain their expectations. Professional UX architects should be adept in the role of acting as a liaison between the project team and stakeholders, but since you're ultimately accountable, you should control the situation. If you involve stakeholders in initial product architecture questions, you'll need to educate them about the purpose and limitations of the initial product architecture stage and make sure their input is properly restrained and consistent with the existing framework requirements.

We will assume that you will be employing professionals in the UX and technical architect roles. Our goal in this chapter is to provide you with an understanding of what goes into UX and technical architecture, to help you better supervise, interpret, and communicate the value, process, and results of this stage.

UX Architecture

UX architecture sheds greater light on the problem, further refines the framework requirements, and defines solutions to the pivotal problems. UX architects do this by looking at the problem through a variety of lenses, and using a number of techniques that are effective at building clarity and suggesting solutions. We use the words "lenses" and "techniques" to highlight the fact that UX architecture, like software development, is not a stepwise process. The organization of this chapter shouldn't be interpreted as an ordered list of steps as in an instruction manual where, if followed precisely, success is guaranteed. It is an overview of techniques and methods of viewing the problem (lenses) that are used by UX professionals to deepen their understanding of the problem and begin to propose aspects of the solution.

Contextual Scenarios

Contextual scenarios describe the product's requirements from the user's perspective through narrative descriptions of the tasks users will undertake to achieve their goals when using the product. They are a sort of storytelling technique that's meant to give a clearer picture of how the product will need to behave and what tasks it will need to support, without enumerating them down

to the tiniest detail. Much as user personas provide a framework for making decisions through inference and empathy, contextual scenarios tell a story in broad strokes, leaving the details to be filled in through inference in the minds of the project team. UX professionals write them by intersecting business goals with user stories, user goals, and other information discovered in user research.

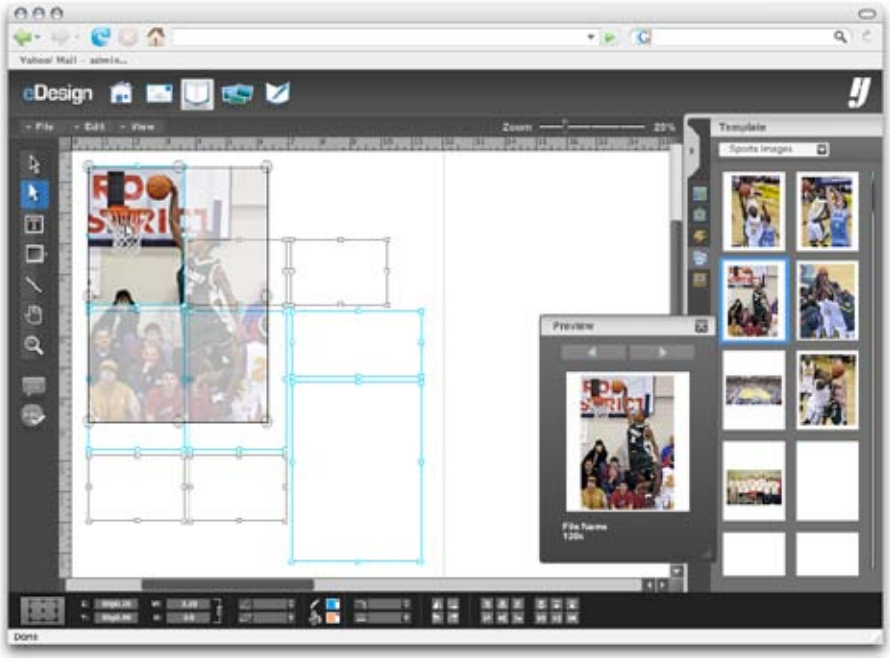
This is a contextual scenario pulled from our work with Herff Jones to produce an online yearbook editing tool:

> *Tina is assembling the homecoming page of the yearbook. She logs in to the site and sees the pages she is assigned to. This makes it easier for her to navigate directly to the homecoming page. It's mostly blank, but the template her class worked on together over the summer has already been applied, so all she needs to do is pick out some photos and arrange them on the page in a fun and creative way. Tina opens the photo browsing panel and sees lots of photos the photographers have taken. She filters the images by "homecoming" and sees about 30 photos that have been tagged that way. Tina clicks on a thumbnail to zoom in to see the image more clearly and pages through the collection of full-size photos. This is Tina's favorite part of working on the yearbook. She selects an image and the photo browsing panel goes away. The image she selected is now on her layout and she resizes and crops it carefully.*

To illustrate why contextual scenarios are a strong means of describing requirements, let's look at just one sentence from this example: "She filters the images by 'homecoming' and sees about 30 photos that have been tagged that way." This sentence alone implies the need for many features and capabilities, including:

- *Photos are digital assets in the system.*
- *Some mechanism for importing digital photos into the system must exist.*
- *Photos can be tagged with properties that describe their subject.*
- *Some mechanism for tagging photos must exist.*
- *There must be some facility for browsing photos.*
- *The photo browsing facility must support filtering of photos based on tags.*

Notice that the first four implied requirements fall outside the view of the user's activities described in the contextual scenario. The photos have been imported and tagged before Tina's activities begin. This demonstrates the power and effect of designing products around the user's perspective; attending to the user's needs implies requirements and functionality that the user might never be aware of or personally encounter.



End result of the "homecoming" scenario

The practice of describing tasks using contextual scenarios that imply but don't specify details is in keeping with the discipline of restraint and the acknowledgment of the weakness of written requirements. UX architects allow decisions about the specifics of the solution to be made during development (when the problem and possible solutions are better understood) by leaving it to the project team to read between the lines from contextual scenarios. By leaving out specifics, it becomes possible to create a form of written requirements that are comprehensive in their breadth and are entirely reliable because they describe only what's known at only the level of detail that's available.

Contextual scenarios are an effective means of elaborating on the framework requirements because they have the trademark characteristics of framework materials:

- *Fixed, reliable, and certain about what's known*
- *Flexible, inclusive, or permissive about what isn't known*

The sample scenario requires, for example, that a mechanism for filtering photos based on tags exists. But the scenario doesn't attempt to specify exactly how filtering will be accomplished, the nature of and constraints on tags, what other activities might also be available through the same photo browsing screen, and so on. The specifics are left to be decided when things are better understood and when specific solutions are more apparent.

Contextual scenarios can be created in storyboard form in addition to textual form. Storyboards are useful in creating an even more emotionally appealing and implication-rich view into the user's life and needs. They also help keep the project team focused on the wider context and environment in which the user is using the product.

Mapping High-Level Workflows

A workflow is a sequence of steps the user will undertake to perform a task or accomplish a goal. Workflows can be high-level (pertaining to major sections or functions of the application) or low-level (pertaining to a specific, narrow task). For example, the high-level workflow for sending an email is something like this:

- *Enter recipients in the "To," "Cc," and/or "Bcc" fields*
- *Enter a message subject in the "Subject" field*
- *Compose a message in the message body editor*
- *Optionally, choose which email account to send the message from*
- *Click "Send"*

Note that each step in this workflow is presented and organized in a single application screen (the message composition window).

The Herff Jones example implies a number of different interconnected workflows. Let's focus on just one part of it:

> *Tina opens the photo browsing panel and sees lots of photos the photographers have taken. She filters the images by "homecoming" and sees about 30 photos that have been tagged that way. Tina clicks on a thumbnail to zoom in to see the image more clearly and pages through the collection of full-size photos.... She selects an image and the photo browsing panel goes away.*

At a high level, this describes the workflow for placing a picture into a yearbook layout. In the email workflow example, all of the workflow steps are presented in a single application screen. In this contextual scenario, however, the need for multiple application screens is implied:

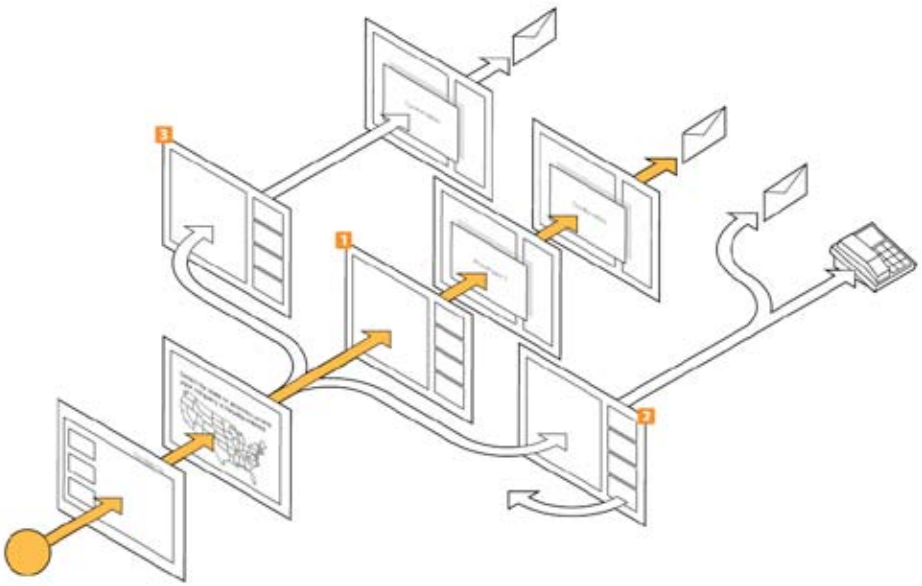
- *A photo browsing "panel" comprising (at least):*
  - *A photo thumbnail viewer*
  - *A text-input filtering mechanism*
  - *The ability to select a photo to view it full-size*
- *A full-size photo view comprising (at least):*
  - *A single photo viewer*
  - *A means of paging through full-size photos*
  - *A means of selecting the image in view for placement in the layout*

So, the high-level workflow implied in the contextual scenario itself implies that certain application screens exist. Because the existence of application screens is implied in a workflow, it's premature and unnecessary to try to figure out the organization of application screens at this stage. So, the job of mapping high-level workflows involves identifying those workflows, figuring out what steps they comprise, and determining an order or organization of the steps that's the easiest and most efficient for the users. Most high-level workflows comprise a number of low-level workflows, too. But unless a low-level workflow is very complicated, innovative, or represents an unusually high degree of uncertainty, initial product architecture is typically only concerned with high-level workflows.

The Herff Jones example shows how contextual scenarios can be very useful. They describe what features need to be available to the user, they imply sequences of tasks users will go through to accomplish goals, and they tell stories that suggest how the functionality of the application needs to be grouped and presented. The story of how a user uses the application should clearly suggest the pathways she'll take through it.

It's useful to document the high-level workflows of the application early so the project team can understand how the application's functionality should be logically organized from the user's perspective. The goal here is to map the workflow from the user's cognitive perspective rather than from a systems design perspective. The UX architects also shouldn't start making guesses about what application screens need to exist or start detailing low-level workflows; this should be done later during development.

The following figure is an example of a high-level workflow that shows a single point of entry and several possible outcomes. The primary path is highlighted.



Sketching Low-Fi Visual Representations of Requirements

A full understanding of how functionality might be exposed to the user can be elusive until you start to visualize it. Although the bulk of the work of building detailed wireframes and mockups of application screens shouldn't occur until development begins, early sketches on a whiteboard or low-fidelity "paper prototypes" can be useful as a technique for deepening the understanding of the problem.

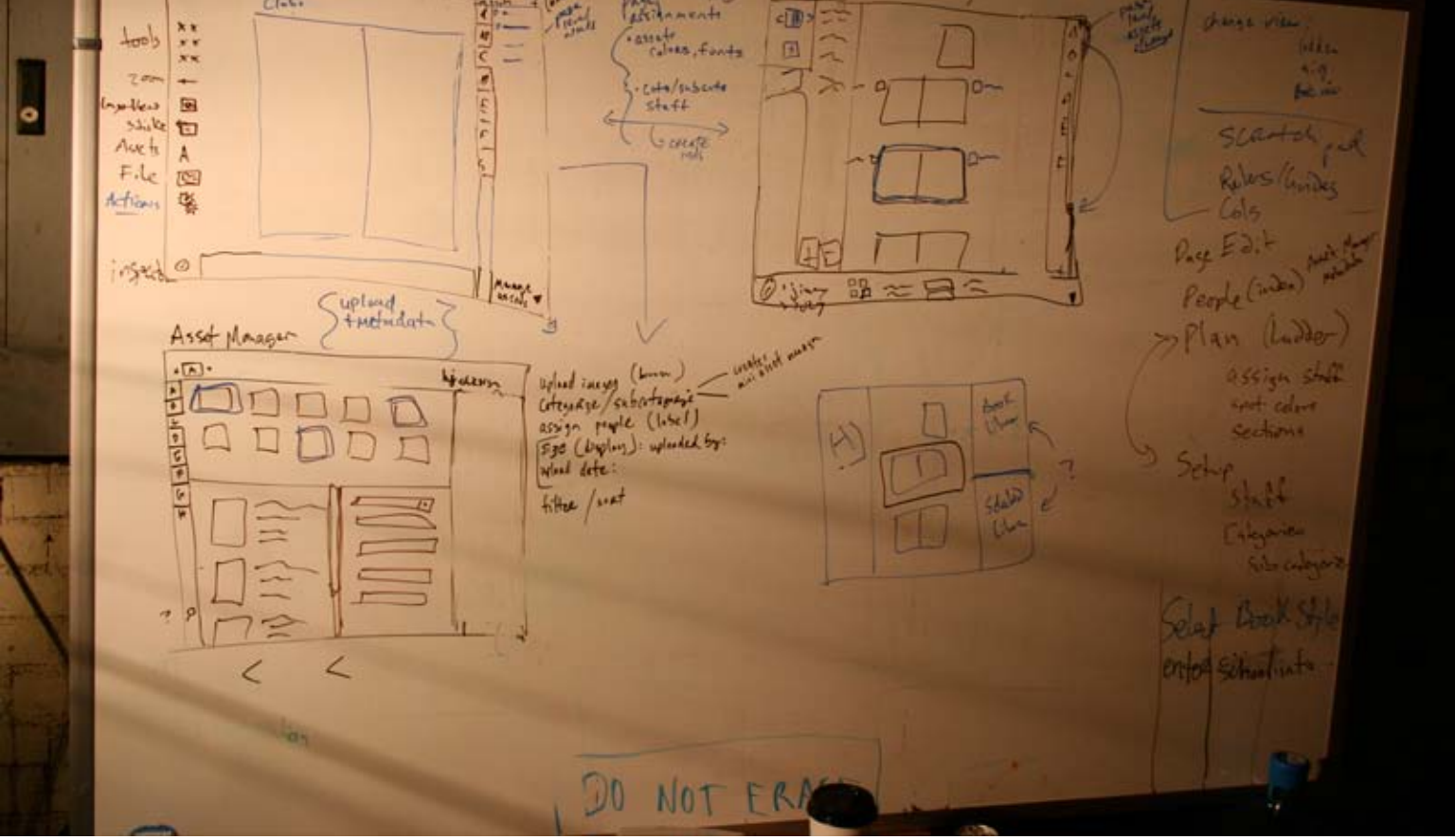
Sketches can be—and often are—simply pen and ink drawings on the back of napkins and on scrap paper. The goal of the sketches isn't to produce detailed requirements or firmly suggest how screens should be organized and composed, so they needn't be detailed, polished, or even accurate. Again, these sketches are simply a technique that can be used to explore and build a better understanding of the requirements.

This picture shows some of the early thinking done for asset management in the Herff Jones eDesign application. There are rough interface elements shown in different arrangements and control clusters shown in different positions on the "screens."

Examining Key Features and Interactions

Though the initial UX architecture stops short of examining and specifying low-level details of the solution, there might be some details that call for early exploration. You might be planning a feature that has never been done before, presents a significant challenge, or that introduces a radically new approach to its workflow or interaction design. You might also be contending with stakeholders who are skeptical or having difficulty picturing how key components of the product will work. Anything that's new, innovative, or challenging is bound to come with more than its fair share of unknowns and risks, and these should be examined during the initial product architecture stage.

The success and viability of the product often depends on finding a good solution to these key problems. You'll want to proceed into development with the confidence that they can be solved within the constraints of the project. To reduce the degree of risk and uncertainty surrounding these problems, UX architects can do a much deeper exploration of the problems and their potential solutions than would ordinarily occur this early in the project. These explorations can take the form of some basic wireframing to illustrate interactions on paper, but might be as complex as a building a working prototype of the feature. Success in an exploration might be in proving the technical feasibility of something, in receiving stakeholder approval and support, or in receiving positive feedback on the feature from sample users. The more risk there is in a given detail, or the more dependent its success is on user acceptance, the more important it is to create a higher-fidelity prototype.



Lo-fi sketches on a whiteboard

Setting a Style Vision

The visual design of a product's UI can have different tones, moods, and stylistic genres depending on the product's audience and the purpose. Some software—educational applications for children, for example—are resplendent with candy-colored interface elements, use happy or goofy text styles, and emphasize fun, simplicity, and accessibility. Products made for professional stock traders tend to have very subdued tones and a relatively austere aesthetic, focusing on effective delivery of information without distractions from the interface design itself.

This doesn't mean that UI design is important for the children's application but unimportant for the stock trader's application. UI design considerations are critical to the experience of using the application, no matter what the intended experience might be. Many people believe that in enterprise or heavily data-focused applications, the UI design needs to "get out of the way" and isn't an important concern. But even in cases that demand an extremely austere UI design approach, the design still significantly affects the subtleties

that create the experience of using the application. Stock traders don't need an application that will entertain them, delight their budding senses, and seize their fickle attention like children do. But they do need to feel that the application is high quality, professional, reliable, and sophisticated. So, one of the goals in initial UX architecture is to set out a mood and style vision for the product that sets the right genre and purpose associations for users.

Like art and fashion, software UI design has distinct genres as well as design trends that change over time. Modern UI design trends are recognizable even to people who aren't actively paying attention. The Web 2.0 trend has been accompanied by its own relatively distinct genre of web design. As long as Web 2.0 is seen as cutting edge, design styles from the Web 2.0 genre will be associated with modern, sophisticated software. We're frequently asked to design interfaces that are "clean" or "airy" or "crisp," using "friendly" UI elements and iconography. Clients requesting this are typically expressing the desire that their application UI look modern and sophisticated, because at some conscious or subconscious level they've noted that those characteristics are present in many of the cool new things. The product UI design is also a means of expressing the brand goals of the product or of the company generally.

It isn't important during UX architecture to lock down the precise color palette, iconography, or other specific elements of the UI design for the product. But it is useful to begin with a general sense of the mood, genre, and experience that the UI design will ultimately need to convey. The attributes of the experience or of the brand that you're trying to create are difficult to express in words. In setting the style vision for the product during UX architecture, vague understandings and expressions of visual ideas can be made concrete. That will give clear direction going forward and ensure stakeholders are all imagining and expecting the same things. This initial style vision will be the framework within which future visual design work is done. It also helps members of the project team visualize what the product will eventually look like so they have an easier time imagining their contributions in context of the whole product.

To begin developing a style vision, UX architects often ask stakeholders to make lists of other products, websites, print advertising, and brand design that stakeholders feel are representative of their style goals for the product. There's rarely an existing product that exactly represents the stakeholders' goals for their own products, but with enough examples, UX architects can get a clear sense of them. UX architects and UI designers are deeply immersed in the genres and design trends of software UI design, so they can readily support stakeholders through this process. They can help stakeholders clearly express subjective concepts, provide illustrative examples of ideas, and work to corral opinions to an outcome that their professional experience suggests is correct.

Based on the suggestions from stakeholders, and using some of their own materials, UX architects and UI designers document a vision of the design goals for the product using what are called mood boards. *Mood boards* are essentially collages of images, colors, and designs pulled from various sources that, in aggregate, give a clear suggestion of the product's design mood, genre, and approximate color palette.

Mood boards can also be a useful tool in getting some early user feedback on the design direction. On the Herff Jones eDesign project, we had internally arrived at a visual direction for the product that was consistent with other professional design applications. The interface was intentionally dark to boost contrast with the lighter content that users would be developing. But we began to worry that this approach might be off-putting to the primary users of this app—teenage girls. We were both right and wrong. The users we tested the visual concepts with appreciated the contrast but needed some deeply saturated colors interjected to maintain their interest. A new set of mood boards that balanced high contrast with very saturated colors in the controls seemed to resonate well right away when we tested it with users.