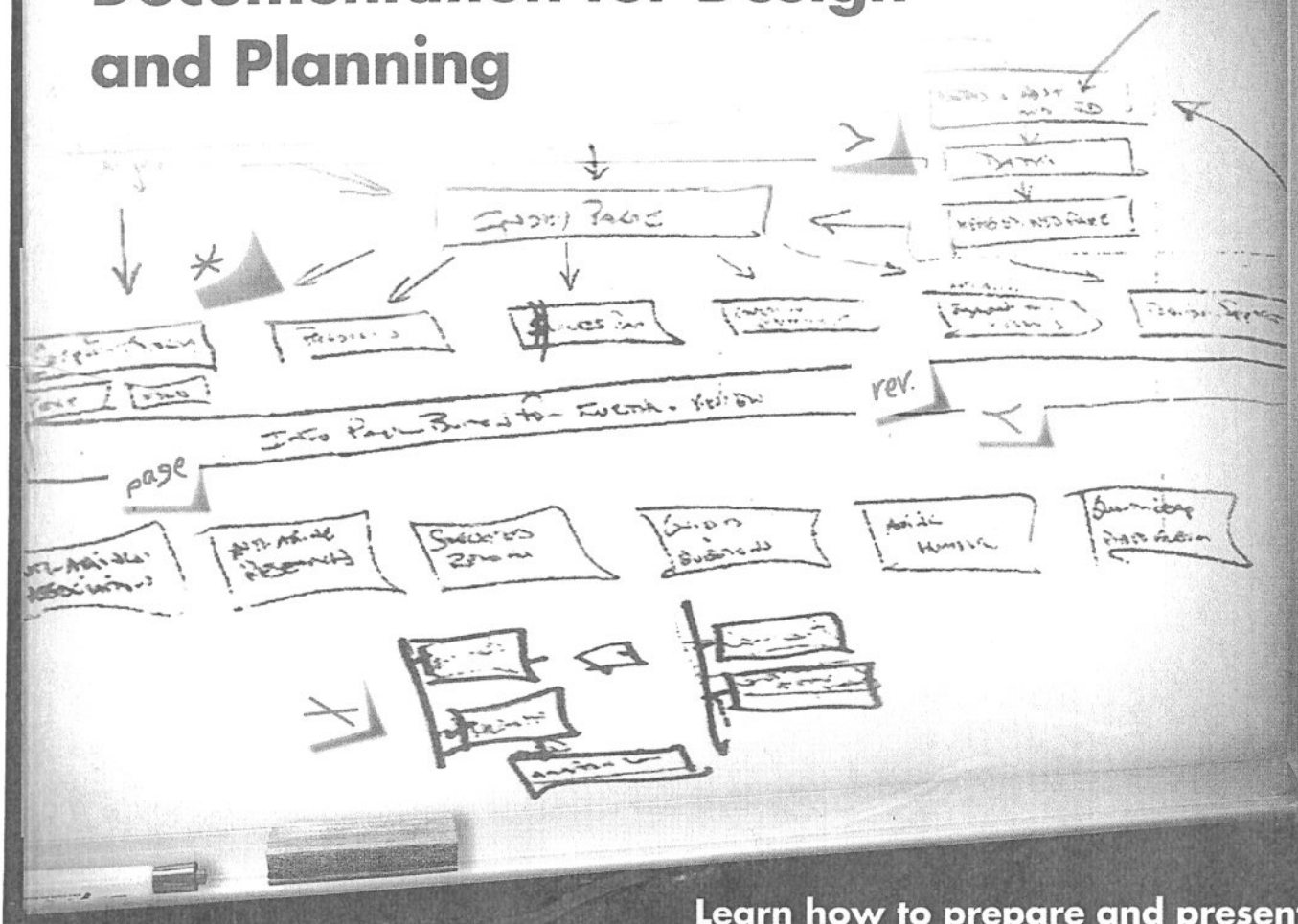
Communicating Design

Developing Web Site Documentation for Design and Planning



Learn how to prepare and present wireframes, site maps, flow charts, and more

New Riders

Dan Brown

CHAPTER TEN

Wireframes

wī-ər-frāmz′ (n.)

A simplified view of what content will appear on each screen of the final product, usually devoid of color, typographical styles, and images. Also known as schematics, blueprints, prototypes

Wireframes are rough illustrations that show, to a greater or lesser extent, the contents of each screen. They're called wireframes because they are typically rendered with simple lines, not elaborate designs. They illustrate, among other things, what kinds of information will be more prominent on which screen. It sounds simple, but wireframes are among the more controversial documents in the user experience library because they blur the line between underlying structure and visual design. In other words, wireframes cross the boundary between structure (how one kind of information relates to another kind) and display (how to represent information on the screen).

Wireframes at a Glance

Every team develops a coping mechanism to harmonize the responsibilities of various team members. But, since it would be impossible to address every nuance of every situation, this chapter captures the approach to wireframes that has worked best for me throughout my career, and makes these assumptions:

- The purpose of a wireframe is to communicate initial design ideas. Therefore, the team must have a solid understanding of the design problem.
- The scope of a wireframe is content and structure, not layout or visual design (though wireframes may be used in deliverables later in the process to document those issues). To a certain extent, wireframes illustrate how users will interact with the web site, but since they're typically presented on paper, they're not the ideal place to do that.

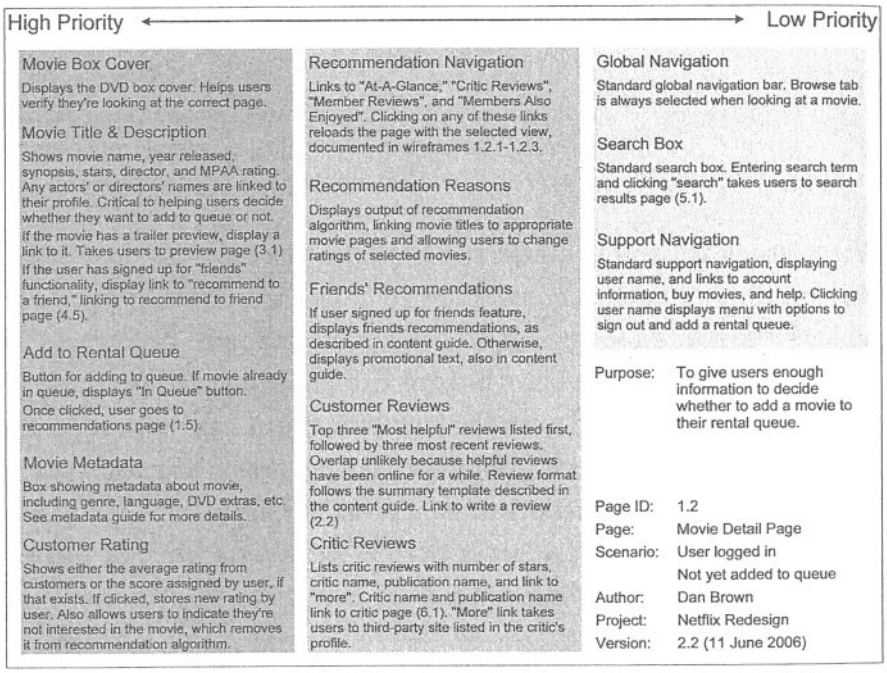


Figure 10.1 This is the simplest wireframe possible a description of the screen's content listed in priority order. To add some depth, this wireframe divides the information into three levels of priority high, medium, and low. Note how different areas of the page have been grouped together in discrete areas. Note also how the wireframe makes no claims about the design of the page, only the structure.

- The primary audience of a wireframe is the rest of the project team—designers and engineers. Wireframes contain information that is essential for the other team members to do their jobs.
- The wireframe must communicate what content the user expects to see on each screen and the relative priorities of the content on that screen. Wireframes are most effective when used in conjunction with other deliverables, since they tell only a partial story.

There are so many different techniques for creating wireframes and using them in a project. In the examples of wireframes below, I've supplemented my own work with some research from the web, in particular the excellent tutorials on the online magazine Boxes & Arrows (www.boxesandarrows.com). Where I've departed from my own experience, I refer to the source material in the text itself. A comprehensive list of sources cited appears at the end of the chapter.

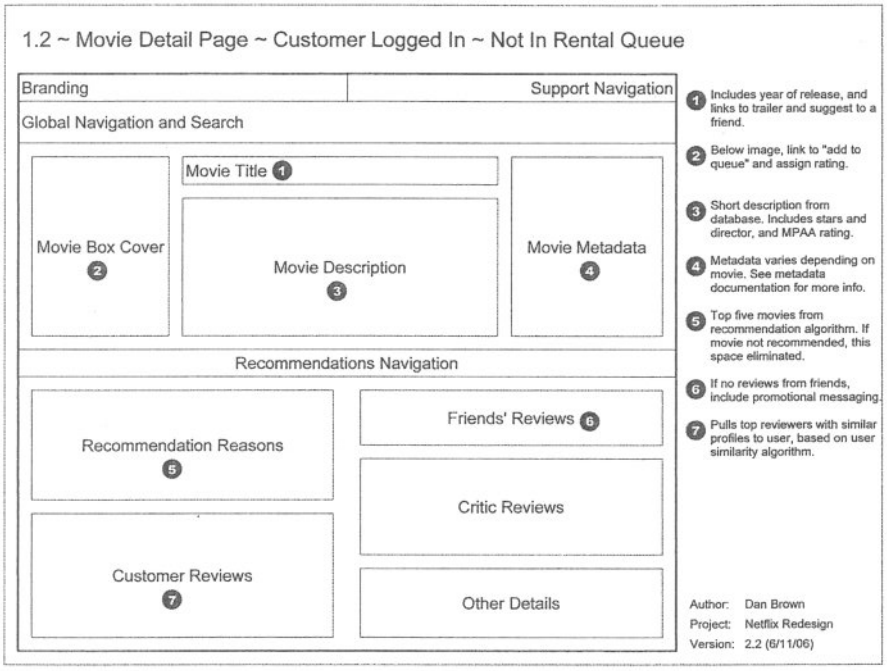


Figure 10.2 Unlike the previous wireframe, this one shows the layout of the page. Because it attempts to show what the page looks like, it would be a little weird to try to describe the content inside the rectangles—content descriptions may require more space than the content itself. The descriptions, therefore, are captured in annotations to the right of the page.

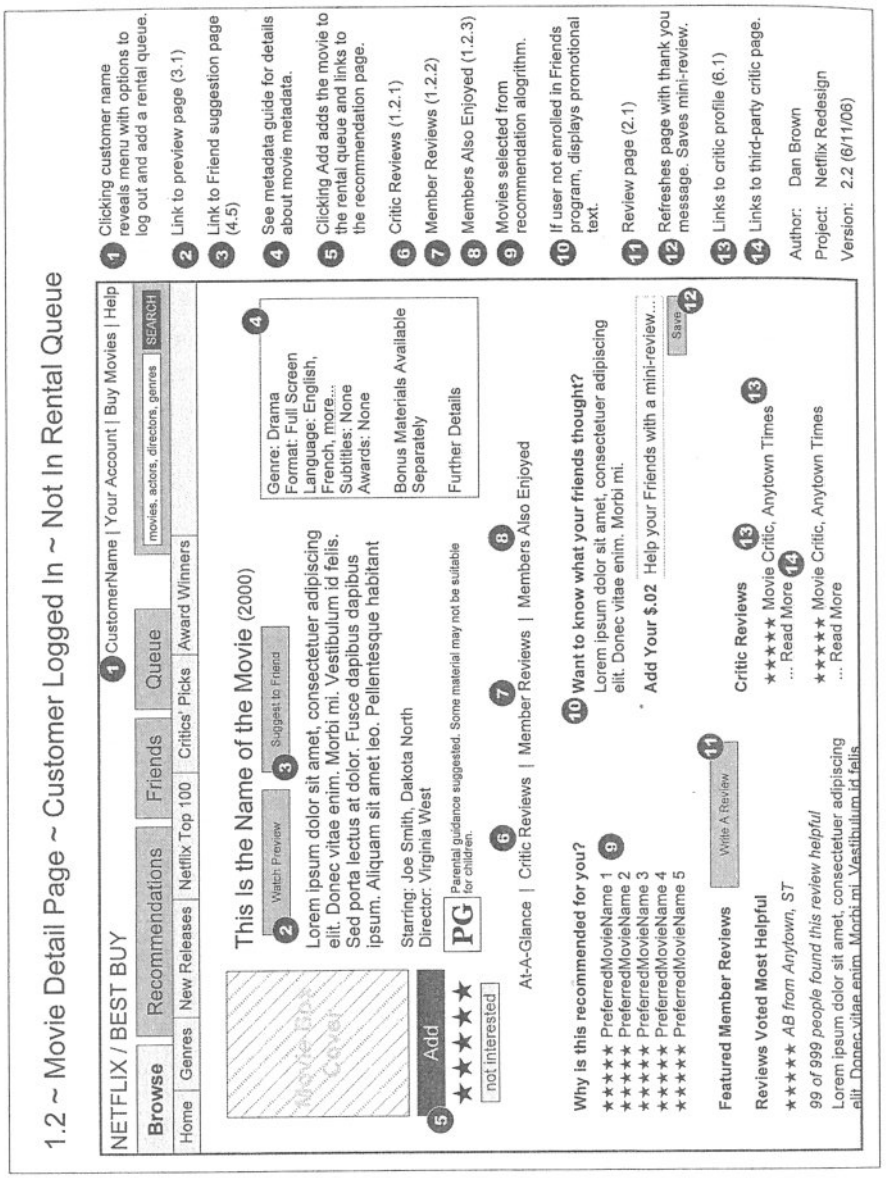


Figure 10.3 Finally, this version of the wireframe matches the screen layout and contents almost perfectly. It is a pared-down version of the final design, but does try to get the proportions correct. Note how the content varies. In some places, the content is exactly what it would be on the actual site (like labels and navigation). In those areas that are more dynamic, the wireframe shows placeholder text (like "The Name of the Movie Goes Here"). The annotations to the right describe links and functionality. This chapter will explore the pros and cons of creating a so-called "high-fidelity" wireframe.

Wireframes

Purpose—What a

Wireframes may b
tural decisions to
usability testing. W
you don't depart f

Audience—Who u

If you're using wi
be the system's au
and structure dec
making the stakel
higher than appro

Scale—How muc

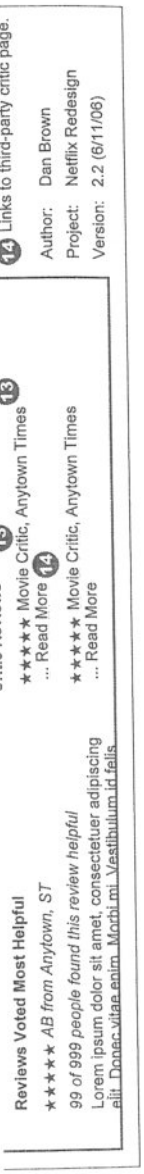
The scale of wire
detail in each wi
wireframes. The l
fidelity" means lo
or "high-level." C

Context—Where

Ideally, wireframe
work—like flow

Format—What d

In the taxonomy
tronic. Of the pa
and labels to rep
complex design
part, wireframes
one page of the
from high-end i
to more mundar
wireframes, som
the contents of
screens may feel
simple HTML
the team's proce



ost perfectly. It
ow the content
nd navigation). In
the Movie Goes
the pros and cons

Wireframes Overview

Purpose—What are wireframes for?

Wireframes may be used for a variety of purposes, from communicating structural decisions to the rest of the design team to serving as paper prototypes in usability testing. Whatever purpose wireframes serve in your process, make sure you don't depart from it. Wireframes fail when they try to do too much.

Audience—Who uses them?

If you're using wireframes as prototypes in usability testing, their audience will be the system's audience. If your wireframes are intended to document design and structure decisions, they will be used by the rest of the design team. Avoid making the stakeholders your primary audience since their expectations may be higher than appropriate for this stage in the project.

Scale—How much work are they?

The scale of wireframes may be measured on two dimensions: the level of detail in each wireframe and the number of screens represented in the set of wireframes. The level of detail is often referred to as "fidelity," where "high-fidelity" means lots of detail, or "low-level," and "low-fidelity" means less detail, or "high-level." Confused? This is just the beginning.

Context—Where do they fall in the process?

Ideally, wireframe creation begins somewhere between high-level structural work—like flow charts or site maps—and screen designs.

Format—What do they look like?

In the taxonomy of wireframes, there are two main species: paper and electronic. Of the paper-based wireframes, some are sparse, using only rectangles and labels to represent different areas of the screen. Some wireframes use more complex design elements, like color, shading, and typography. For the most part, wireframes are delivered as a set, with each page of the site described on one page of the wireframes. The tools to create wireframes for printing range from high-end illustration programs like Adobe Illustrator and Microsoft Visio to more mundane applications like Microsoft's PowerPoint. For electronic wireframes, some design teams use simple, unformatted HTML to represent the contents of each screen. Often, the experience of looking at these HTML screens may feel a lot like looking their paper-based cousins. Whether these simple HTML documents get integrated into the production system is up to the team's process, but anecdotal evidence shows it rarely saves any time.

Challenges

Even five paragraphs won't do justice to the challenges of wireframes, and ultimately you must weigh the benefit of wireframes—as a potentially useful initial representation of the user experience—with the costs, which I'll discuss below.

Web design teams see wireframes as a panacea—an opportunity to kill many birds with one massive boulder. With wireframes, you could legitimately document functionality, present initial screen design concepts, conduct user testing, experiment with different interaction models, elicit requirements, validate requirements, prioritize content, review draft copy in context, and much, much more! Plus, if you act now, we'll throw in strained team dynamics, stakeholder waffling, and scope creep absolutely free. (Make your checks payable to the author of this book.)

Good documents have singular purpose, and the closer they remain to that purpose, the more effective they are. With complex projects, the temptation is to stretch your documents to do more. But the more you try to do with wireframes—or any document or tool, for that matter—the less effective they will be in all those things.

For one government agency, a team created wireframes that attempted to show content priorities, production copy, links, and navigation—ultimately, the site itself rendered in Microsoft Visio. The document was a mess because it had to serve four different audiences: the client, the copywriters, the graphic design team, and the HTML jockeys. It was difficult to keep the wireframe up-to-date because there were so many people commenting on it. Although the wireframes were a central place for capturing all decisions, they did not do a great job, no doubt owing to the strain of competing priorities combined with the limitations of the medium.

This chapter on wireframes appears in the "what users see" section of the book because, based on the assumptions above, wireframes are best suited to capturing a design team's ideas on how a site should behave. That is, the design team already has an idea of what the site is supposed to accomplish and the wireframes offer a first stab at how it accomplishes its purpose. The challenge is that wireframes can be used for other aspects of the overall process, as well—not just to show design, but to better understand the problem at hand.

If you take nothing else from this chapter, take this: Decide upon a purpose for your wireframes and stick to your guns. Once you know what the wireframes

on your project are for, you can design them to serve that purpose and set the project team's expectations, hopefully avoiding wireframes that ultimately do nothing for nobody.

Creating Wireframes

Typical wireframe documents have pages that are divided into two areas. The larger area on the page has the wireframe—the depiction of what's on a particular page—itself and the smaller area has supporting information that sets the context and describes the behaviors of the screen being displayed.

Like all the documents in this book, wireframes can be described in a series of three layers. The first layer contains the essential content, in other words, the elements of the document that make it what it is. The second and third layers add increasing levels of detail that are more or less optional, depending on your situation.

Layer 1: Wireframe Essentials

The first three elements described in the first layer appear in the wireframe itself—the visual representation of the page. The second two elements, the identifying information and the administrative information, appear in the area for supporting content.

Content areas

Most web pages can be divided into areas of content—discrete rectangles that hold different types of information. Sometimes a page has just one content area and sometimes it has many. The content area is the basic unit of currency for web pages.

A wireframe can represent these content areas literally, showing a large rectangle (the page) divided into smaller rectangles (the content areas). These content areas don't necessarily show actual content—they may only show a label or a sentence describing the content. (More on this shortly.) The wireframe could approximate the screen layout, so the rectangles reflect the actual screen real estate dedicated to each content area. While this may be useful to give project participants a sense of what the system will look like, it can stifle creativity in later phases of the project, when the team is working on the actual screen design.

Another approach is having the size and position of the rectangles reflect relative priority of different items, rather than the actual page layout. The value of this approach is that it provides direction to other members of the team without prescribing a particular layout.

One other possibility is to abandon any semblance of rectangles or screen layout and simply list content areas in order of priority. Such an approach significantly departs from the typical wireframe, but if your team is experiencing conflict due to the squishy nature of wireframes, this technique may help.

Content descriptions

It's not enough to simply indicate that a page is made up of different content areas. The wireframe must describe the content that will appear in each area. There are several techniques for doing so: using actual content, some form of sample content (described later), or a short description of the content. At a minimum, the description consists of a label like "list of articles" or "application form." The simplest wireframes are groups of rectangles with labels defining content areas. More complex wireframes include real content—everything from heading labels to navigation to actual prose.

Content priorities

One goal of any kind of visual design is to establish a hierarchy of importance among the content on the screen: On a good web page, for example, it's immediately clear that some pieces of information are more important or pertinent than others. More important content must be made salient in some way, either by giving it more screen real estate, contrast, or some other graphic design technique. Less important elements may find themselves on the periphery, represented with smaller typefaces, or rendered in some other way to avoid distracting people from the main purpose of the screen.

The purpose of the wireframe is to rank content in a simple, clear way, establishing guidelines for those who will design the site. Sometimes the relative priorities can be represented as a simple list, with the most important item at the top. In many cases, however, the relationships between content and functional elements are much more complex. To accommodate these complexities you can group related items in your priority list.

Wireframes are an intermediate step in the design process—they help designers understand all the information that each screen will need to accommodate.

Having this blueprint helps the design team ensure that they're not only meeting the needs of the users, but also the needs of the content.

Identifying information

Because wireframes tend to appear one to a page, and because they're generally presented on paper, each wireframe must include identifying information that answers some basic questions, like "what project is this for?" and "what's the name of this page?" In some cases, a single page may have multiple views—slight variations on the page depending on circumstances. Today's web sites are highly interactive—the user can accomplish a lot without appearing to leave the screen. In these more advanced applications, a single screen can have multiple views. For example, a site's log-in screen may have three different views depending on the scenario: default, username not found, and password error. From the user's perspective the screen remains pretty much the same, with only slight variations. The wireframe must show all three views. Even though these might be considered one "page" on the site, you might represent it with three different wireframes—one for each view.

The wireframe's supporting information, therefore, should indicate which page and which view of the page.

You may also have a system for enumerating your wireframes (e.g., 1.1, 2.4.1, etc.). Some project teams find these useful and others just find them difficult to follow. If your wireframes need to correspond with a site map or flow chart, a numbering system can help with the bookkeeping. On the other hand, if the site structure undergoes dramatic changes during the design process, you may find keeping these identifiers up to date more trouble than it's worth.

Administrative information

In addition to identifying the wireframe in the context of the system, the document must identify the wireframe in the context of the overall project. It's easy to forget to add the author's name, the page number, and the version number, but now you have this book and will never forget again. These elements should appear on every document described in this book.

Layer 2: Filling in the Story

Wireframes with just elements from layer 1 will get the job done, but they may benefit from some of the elements from layer 2, which provide additional details

about how the screens function. Figure 10.3 at the beginning of this chapter contains some of the elements described here, like annotations.

Scenarios

Though it may not be essential in most cases, in some documents identifying a scenario for each screen may help explain its purpose. The scenario describes the situation that brings the user to this particular screen.

Scenarios, for example, may account for slight differences between otherwise similar screens. Some web sites—intranets in particular—may display different information depending on the user. Managers or administrators may have permission to view different kinds of information, for example, that regular employees don't have access to. Putting a scenario on your wireframe helps readers answer the question, "Why would I see this version of the screen vs. that version?"

This may sound a lot like a view, described above as "slight variations on the screen depending on circumstances." In some ways they are redundant: A single screen may have several different appearances, depending on some variables. The distinction, however, is in the source of the variation. The view gives a name to the variation and the scenario explains why the variation exists. A view is specific to a particular screen, and a scenario can affect any number of screens. In other words, the fact that a user forgot her password is relevant only to the log-in screen, but that she's a system administrator has implications across the entire site.

Links and form elements

To make your wireframes more detailed, you may elect to show how users interact with the site by indicating links and form elements—things the user can click on and type into. By including these, you are further illustrating the user experience, adding a dimension to explain the interaction between user and system. When they appear on your wireframes, functional elements raise the question: "What happens when the user...?" Consider coupling links and form elements on your wireframes with annotations to describe what happens when the user interacts with them.

Because a wireframe is like a screen without any stylistic information, links should appear in their default state: underlined and blue. Typically, blue is the only color used in the wireframe itself to indicate links. (Some purists believe that any color in a wireframe is deliverable treason.) Since underlined blue text

is near-universal representation of actionable text, it effectively differentiates functional elements from nonfunctional.

As for form elements, these should also appear in their unstyled state: simple gray boxes.

Annotations

Showing the content of the screen is not enough, especially in modern web sites offering high degrees of interactivity. A highly interactive web site has depth—a conversation between user and system—which is difficult to represent in the two dimensions of a wireframe. Annotations remedy the inadequacies of paper by explaining the interactivity of a page in a short note. Some annotations describe the functionality of an element on the page, some explain the rationale behind it, and others provide further description or direction for content. With a dozen or more annotations on one screen, it's enough to give you a headache. You can format these different kinds of annotations differently to draw the attention of different audiences. Developers will be more interested in functional annotations, for example, so it is useful to have a way of distinguishing those from the others.

Functional annotations describe what happens when the user does something—clicks a link or a button, in most cases. The annotation can provide some description of the system's response to that action, like "The system validates the address information, comparing ZIP Code to City and State, and confirms that all required information was supplied. If not, the system returns the error screen (1.1). Otherwise, the next screen is displayed (2.0)." Of course, if your audience is familiar with how various screens work, you may get away with simply noting the reference number (as in the example above) or a URL.

Content annotations either provide direction to your team's copywriter or describe the source for the content. Since you're not providing the content itself in the wireframe, your team might benefit from further explanation of the kind of content you expect to appear in a particular area. If the content is dynamic, drawn from a database, you might explain the rules for pulling that content. If your team includes copywriters, these annotations save you from having to include sample copy. These annotations may simply indicate the gist of the content area, like "Describes the overall checkout process" or "Indicates the product's limitations."

Finally, the wireframe may capture the rationale for particular design decisions. Although you would no doubt cover these when you present the wireframes,

documenting them can ensure that people who missed the presentation will understand the rationale. Don't underestimate the power of rationalizing your design decisions. Later in the project you may forget why you placed certain content on the page.

If an annotation refers to a specific element on the screen, place a numbered target next to that element. The annotation should appear with the same number in the sidebar. To distinguish different kinds of annotations, you can use different colored targets. But keep in mind that the target color should be something other than blue or red, which are generally used in the wireframe itself to indicate links and error messages.

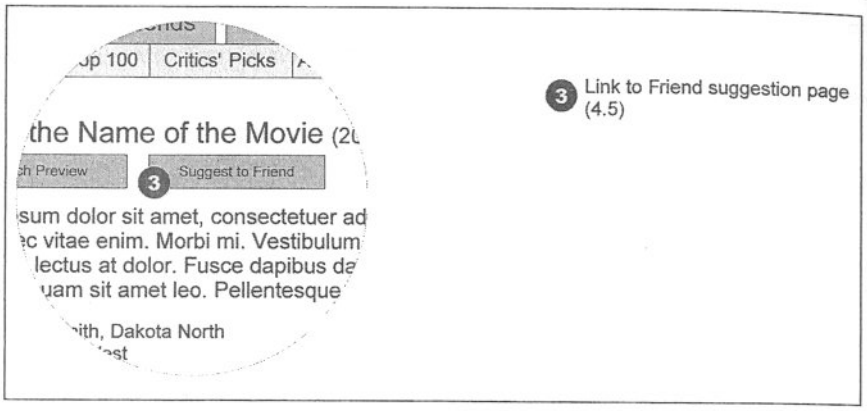


Figure 10.4 Annotations are useful for describing the behavior of interactive screen elements. They're usually marked by corresponding numbered circles—one in the wireframe pointing to the functional element and one next to the annotation itself.

Another approach is simply to treat all your annotations the same, whether they relate to function, content, or rationale. The annotations will have different numbers, but will be formatted the same way. This approach eliminates a data point—that is, the kind of annotation for a given screen element—but that level of detail may not be necessary for your purposes. This approach may also make sense when there's a lot to say about only a few elements on the page.

Objectives and rationale

While annotations refer to specific content areas of a screen, objectives and rationale generally describe the *entire* screen. Although the sidebar on a wireframe

will be dominated by identifying information, administrative information, and various annotations, your situation may require some additional context, and the sidebar is a good place to put it. You may want to set the stage for a wireframe by indicating the screen's objectives—for example, its role in the system, the overall rationale for including it, or the philosophy behind its design. Figure 10.2 includes an example of integrating a purpose statement into a wireframe. Besides reminding team members about the context, this information can help contain the conversation, focusing it on relevant feedback for the situation.

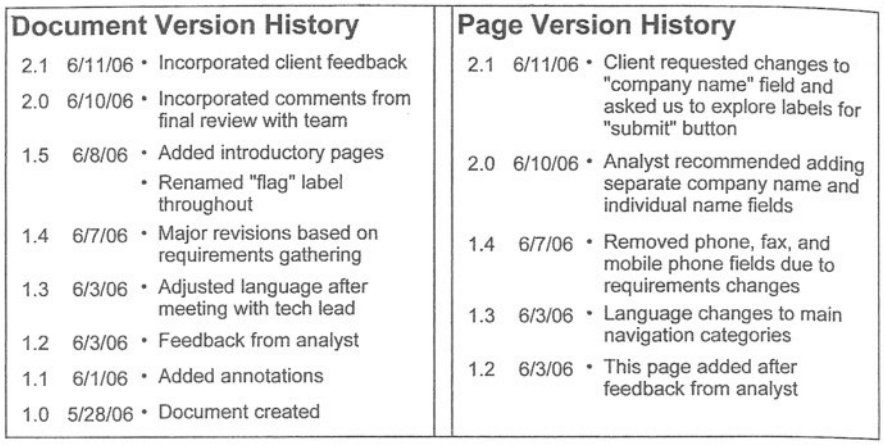
Now that you've indicated what user scenarios the screen is meant to address and the design objectives for the screen, you may need to spell out the rationale. Rationale connects design decisions to objectives or scenarios, showing how the design was derived from the given direction or requirements. An absence of rationale can be telling, because it could suggest that design decisions are without basis. Some of the content descriptions in Figure 10.2 also include rationale, usually to explain the prioritization of certain elements over others.

Version history

A version history accounts for all the changes that have been made to a document, at varying levels of detail. Some version histories note every minute change while others describe changes at a broad level. Not all wireframes are created equal, however, even within the same deliverable. Some wireframes, for better or worse, will get more attention than others, simply because they represent more important screens or have more complex functionality. These screens are likely to be updated more often than the less-important screens. To address this inherent wireframe asymmetry, it may be useful to attach version histories to each screen, in addition to the document as a whole.

Another reason to provide a detailed, screen-specific version history on each page is that it may simplify matters in the final, frenzied stages of the process when it becomes necessary to track the evolution of each page. On these individual screen version histories you can note field label changes, content direction changes, link changes, behavioral changes, etc.

Version history can take up a lot of space, and you'll need to decide how important it is relative to the other information included in your wireframe documentation. If you're keeping each version in a different file, one way to avoid taking up valuable real estate is to note only the changes from the last version.



Document Version History

2.1	6/11/06	• Incorporated client feedback
2.0	6/10/06	• Incorporated comments from final review with team
1.5	6/8/06	• Added introductory pages • Renamed "flag" label throughout
1.4	6/7/06	• Major revisions based on requirements gathering
1.3	6/3/06	• Adjusted language after meeting with tech lead
1.2	6/3/06	• Feedback from analyst
1.1	6/1/06	• Added annotations
1.0	5/28/06	• Document created

Page Version History

2.1	6/11/06	• Client requested changes to "company name" field and asked us to explore labels for "submit" button
2.0	6/10/06	• Analyst recommended adding separate company name and individual name fields
1.4	6/7/06	• Removed phone, fax, and mobile phone fields due to requirements changes
1.3	6/3/06	• Language changes to main navigation categories
1.2	6/3/06	• This page added after feedback from analyst

Figure 10.5 Compare the document's version history with the screen's version history. The document's history points readers to pages that have undergone changes since the last version, saving the specifics for each screen. This keeps the document version history relatively lightweight. It also puts the change details where they matter most—on the screen itself.

Layer 3: Optional Details

The elements that appear on the third layer are not only optional, they're controversial. If you want to see a good flame war, pick your favorite user experience online community and ask whether any of these elements should be present in wireframes.

There are pros and cons to including these elements, but remember that nearly every project team and circumstance is different. What works in one project may not work in another. Although this book will explore the pros and cons for each of these elements, their inclusion should be guided by only one thing: the purpose of the document in the context of your project. Have a highly collaborative team? Perhaps you can all work together to define the layout of the screens in the wireframes. Have clients that will react negatively to any content out of place? Perhaps you should leave out sample content.

Layout and visual design

Layout and visual design are not crucial to the success of wireframes. This is true, however, only in the idealistic view of system design, where design decisions are made in a logical order, and where stakeholders understand that before committing to a particular layout, you first need to identify what appears on each screen and what's most important. When you find yourself in this situation, feel free to dance for joy, mark the date on your calendar, and do similar dances every year on that day.

In many cases, you will need to show layout and design in your wireframes, perhaps because any other representation of content priorities is too abstract for your stakeholders or because you're working under an accelerated project schedule where you need to make as many decisions as possible in the shortest period of time. Whatever the reason, you'll need to decide just how much to show, and this can depend on the situation, as I'll describe later.

In general, however, the key decision is accuracy: the extent to which you expect the layout of a wireframe to reflect the layout of the actual page. Visual treatment is not an either/or element: You can have more visual accuracy or less of it. Some wireframes are rendered to be as accurate as possible, down to the type treatment. Others are mere guidelines for the design.

TABLE 10.1

Translating Design Elements to Wireframes

Design element	What it is	How it appears in the wireframes
Color	Color is a powerful design element that causes specific and subtle responses, which can vary from person to person.	Few wireframes contain color. It's generally agreed that color is premature at this stage because it causes such a basic response that it easily distracts from the matter at hand: what's on the screen.
Layout	Layout represents information priorities by creating visual relationships between different types of content.	It's nearly impossible to create a wireframe without layout simply because any document has some kind of layout, planned or otherwise. The real issue is whether the layout in the wireframe is meant to simulate what users will see on the screen, or not.
Typography	The treatment of the type—the font, the size, the style—make up the design's typography.	Some wireframes use no typographical treatments whatsoever, employing the same font, size, and style throughout. Others will vary the size to represent different priorities. High-fidelity wireframes—those that seek to capture the screen design accurately—will include full type treatments.

Figure 10.6 Design impacts more than just the entire layout of the page. You'll have to make design decisions about the details of a content area if you're producing a high-fidelity wireframe. In this simple content area from the high-fidelity wireframe at the beginning of the chapter, there are a lot of decisions to make about formatting the individual data elements.

Context in the overall design

The experience context shows how one piece of the experience—in the case of wireframes, a single screen—relates to the whole. You can easily set the context of the wireframe by showing a miniature site map or flow chart with the appropriate rectangle—the one representing the current wireframe—highlighted. Using a miniature version of the site map or flow chart helps stakeholders position the wireframe in the system's entire experience. This technique is only worthwhile, however, if the site map or flow chart has become entrenched; otherwise the miniature version has no meaning.

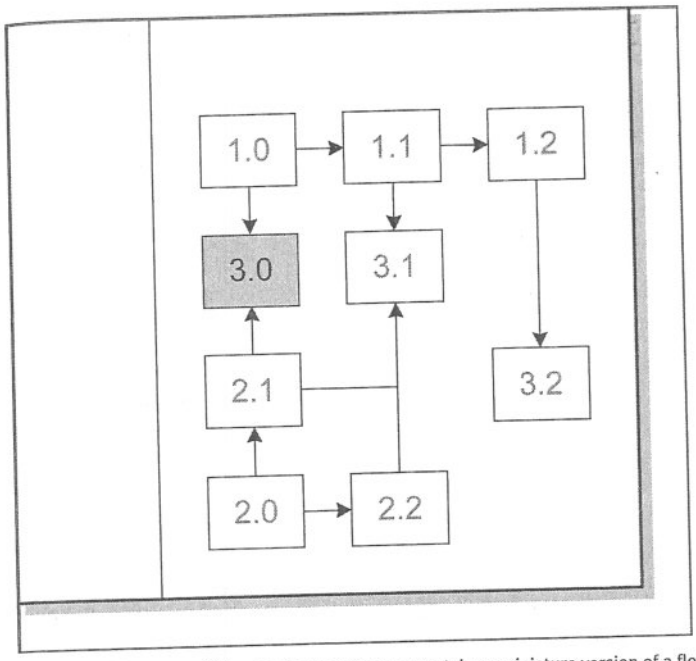


Figure 10.7 This corner of the wireframe document contains a miniature version of a flow chart. The highlighted rectangle indicates which wireframe is being described. If your team is already working with a site map or flow chart, including a miniature version of it with each wireframe can help show the screen's place in the whole system.

Sample content

Sample content is any representation of the content beyond its title, description, and relative priority to other content on the screen. Any time you try to show the content in some way, rather than simply describe it, you are using sample content.

Table 10.2 shows two approaches to the same content. The first approach is the actual content and the second approach is a description of the content. Since at this stage in the project you probably won't have final content—and you won't want to spend your wireframes meeting quibbling over prose—the description is preferable. Like content area priorities giving designers direction about the page layout, content descriptions can give copywriters direction about the subject and tone of particular text on the web page.

TABLE 10.2

Actual Content vs. Content Descriptions

Actual	Description
"By registering, you'll enjoy full membership benefits, including speedy service and 24-hour response to customer service inquiries. You'll have access to your entire order history and be able to set your preferences. Of course, we keep your registration totally private and won't share your information with anyone without your permission."	Explains benefits of registration while reassuring privacy.

Wireframes help establish the organization and relative priorities of content, not the actual wording or tone of it. For that reason, you only need to describe content, not quote it directly. Sample content is a layer 3 element because descriptions of content—a layer 1 element—should be sufficient for the kinds of decisions documented in wireframes. In order to prioritize a content area, in other words, you do not need to know what the actual content is, just what it could be. Like any element beyond layer 1, the decision to include sample content rests on the extent to which it needs to be discussed. In some instances, sample content, regardless of how it is presented, can distract stakeholders from the decisions at hand.

If you feel like sample content is essential for making decisions about priorities, then the wireframe can include it. With some audiences, it can be more difficult to explain why a wireframe doesn't include any content. Maybe putting sample content in your wireframes is worth the risk of having to curtail those conversations. At the same time, it might be just as easy to present a copy deck—a deliverable including all the site's content—at the same time you present wireframes.

There are several different kinds of sample content, from actual content that is prepared especially for the web site, to "greeked" text meant to look like prose. We'll discuss different sample content strategies later in the chapter.

Building Your Wireframes: The Basics

Of all the documents described in this book, wireframes may be the biggest. They have the potential to contain an enormous amount of detail and demand lots of attention from the rest of the project team. In this way, wireframes are a project in and of themselves, and require careful planning.

Prior to starting wireframes, you'll need to make several crucial decisions about the level of detail you'll show for each screen, the standards you'll use throughout the document, and the context you'll present alongside the wireframes. Only a situation analysis—an understanding of how the wireframes will be used—will help you make these decisions.

Timeline: When wireframes happen

As much as we like to think of ourselves as strict adherents to process, we have all fallen victim to the temptation to start designing screens when we have only a partial understanding of what needs designing or what users need. You may have the whole site planned out in your head, even on paper, but there is a time and place to show that work.

The first step in building your wireframes is assessing what you'll want to do with them—in other words, the situation where they'll be most useful. In assessing the situation, you are determining how much detail you want to put into the wireframes.

In some situations, you may be more concerned about the priority of content throughout the site, in which case you perhaps need only a handful of representative screens. If, on the other hand, you're ready to give the stakeholders an overview of the end-to-end user experience, your wireframes may have some detail, but not complete details. Another possible scenario is that your wireframes will serve as highly detailed documentation for the system engineers, which require substantial definition rules and functionality.

Audience: Who's reading your wireframes

If the situation defines the *amount* of detail, your audience will define the *kind* of detail. Engineers, for example, may want to know which database holds the content or field limits or what happens to certain areas of the screen in particular scenarios. Visual designers will be more focused on the details of the content itself, and how much text they need to accommodate in the screen layout.

Purpose: The role of wireframes

Knowing where you are in the project and whom the deliverable is for is not enough. You must also determine the need, answering the question, "What should this document accomplish?" For most deliverables, the need driving their creation is unique and obvious. For wireframes, this is not the case. Because

wireframes can show so much, they are used for a variety of purposes. But remember: Wireframes fail when they try to do too much.

Like any other deliverable in this book, unless you can identify a real need for wireframes, you should not produce them. Still, if you find yourself in a situation where you are required to produce wireframes even though they are not the most appropriate, this document does permit a great deal of flexibility. Identify issues that need clarification—particular business rules or content priorities, for example—and build a deliverable that will drive toward addressing them.

We can boil wireframes down into two possible purposes: to describe a design that addresses a particular set of business needs and user goals, and to facilitate the understanding of business needs and user goals. Logically speaking, these are mutually exclusive: Describing a solution to a problem is different from describing the problem itself. This is a web development project, though, and who are we kidding? Logic need not apply.

When wireframes are used as a design tool they help determine how a site's design will solve a particular problem. Design begins when the designer has an understanding of that problem (for example, knowing what information people need in order to buy a pair of trousers from Gap.com or to manage their money at a banking site) and can devise a product that will successfully address it. Design tools express some aspect of the design and allow the project team and stakeholders to determine whether the approach addresses the problem, as it's been defined in the requirements.

Wireframes can also be used to help make sense of the problem at hand—a process called "requirements gathering." Unfortunately, sometimes you have no idea what the real problem is until you see a solution. By showing a potential solution, the design team can zero in on what, exactly, users need. It's a subtle but important difference. In the first way, you are trying to represent a solution to the problem. In the second way, you are trying to clarify the problem by showing potential (though perhaps inappropriate) solutions. In the second approach, you're not trying to get the solution right. Instead, you're trying to create a discussion piece that can help the project team refine its understanding of the problem.

This iterative approach to design—clarifying your understanding of the problem and revising the potential solution—can work well, and wireframes are suited to it. (One important risk to consider is how much time you put into developing these wireframes. It may be tempting to flesh out every aspect of the design, but without a thorough understanding of the problem, this may be wasted effort.)

Reality frequently dictates a mix of both situations, one of those not-so-rare occasions when we face a catch-22. Though requirements may be established, for example, they cannot be validated until stakeholders get their hands on initial designs. Some newer software development methods try to address this paradox (with faster design processes and shorter engineering phases, for example) but the jury is still out on whether these are effective tools for all types of projects and for creating user-centered products.

Wireframe contents: The nuts and bolts

Having established your situation, you are in a much better position for determining what goes in your wireframes. There are several parts to this decision, and you should try to answer all these questions before putting pen to paper.

How many screens will you show? Though it may be better to err on the high side—to show every permutation of every screen in the site—you need to make this decision based on what role the wireframes play. To simply sell an idea, a handful of essential screens may be all that is necessary. To prepare a complete set of documentation that describes the entire user experience, you may need dozens. For most teams, the wireframes constitute a substantial bulk of the design work—short of the design itself—and therefore require "decks" of between 50 and 100 pages, depending on the complexity of the site.

Answering this question is easiest if you've already prepared and presented a site map or flow chart because you've already discussed the pages with the stakeholders and project team. The decision about which pages to wireframe is a strategic one: You need to balance the need to document the pages with the time and effort it takes to create the wireframes, accounting for the level of detail you need to show.

How much detail will you show for each screen? You may decide to show more detail for screens that play a larger role in the overall user experience, and less on more minor screens.

Once you have a list of screens, identify the various views for each screen. Until you know the relationships between views, however, you may not be able to determine which views you'll need to document. For example, you may decide that two views that are practically identical and therefore do not need their own wireframes can be instead described with annotations on a single wireframe. For each view, determine what content you need to document. Only after listing the content displayed in each view can you distinguish between views that need their own wireframes and those that can be consolidated. This inventory

of screens and views also permits you to establish a scheme for numbering and identifying each wireframe.

How will you represent content? To a greater or lesser extent, wireframes need to describe the content that users will see on the screen. Annotations allow you to include literal content descriptions (for example, "Use this space for a short product description with a link at the end to a longer product description.") But some wireframes include sample content. There are five kinds of sample content that can appear in a wireframe, and each is best suited for a different kind of content.

TABLE 10.3

Different Kinds of Sample Content

Sample Content	What it looks like	Use for	Don't use for	Address rendered
Actual content	This is whatever you have on hand that's as close to production content and data as possible.	Navigation labels, functional labels—in other words, those parts of the interface that the user interacts with.	Marketing copy, tabular data—Using actual copy or data in a wireframe risks changing the conversation to the content of the wireframe, and not the structure.	Peachpit Press, 1249 Eighth Street, Berkeley, CA 94710
Dummy content	Invented content that looks like actual content.	Content with obvious formats (like addresses or phone numbers) such that readers recognize the purpose of the content without getting sucked into the content itself.	Copy or tabular data—It's easy to mistake these for actual data and sidetrack the conversation. To fill a table, you may need to repeat dummy content, or invent lots of different dummy content—both of which are distracting and time-consuming to the reader.	John Doe, 123 Main Street, Anytown, ST 98765
Symbolic content	Strings of repeated letters or numbers to represent different fields. For example, 999-999-9999 for a phone number and MM/DD/CCYY for a date.	Tabular data, dates, information with recognizable formats (like phone numbers).	Prose, unless you want your entire wireframe to look like it's been redacted by the CIA.	XXXXXXXXX XXXXXXXX, XXXXXXXX XXXXXXXXXXXXX, XXXXXXX, XX, 99999

TABLE 10.3 *Continued*

Different Kinds of Sample Content

Sample Content	What it looks like	Use for	Don't use for	Address rendered
Labels	Usually enclosed in brackets, a label provides a description for a field of content. Labels can include additional information about the field, like size and type. For example, [firstname—30] might represent a field that contains someone's first name and is limited to 30 characters.	Pretty much anything.	Labels are nice and flexible and should be able to cover most sample content needs. The downside is that they may be difficult for a nontechnical person to interpret.	[FirstName] [LastName], [StreetNumber] [StreetName], [City] [State] [ZIP]
Greek/Latin	Borrowed from graphic design, prose can be faked by using strings of pseudo-Latin text. (Some people call this "greeked" text and some people call it "latin," while others call it "lorem ipsum" or "lipsum" after the Latin words that typically come first.)	Anything that's typically rendered in more-or-less full sentences.	Lipsum in tabular data or to represent an address just looks silly.	Lorem Ipsum, dolorsit amet lorem, consectecur, st, 99999

What supporting information will you include for each screen? As described in the layer 1 elements above, there are several different kinds of supporting information you can show for each screen, describing its rationale and its position in the overall user experience. You can narrow down what to include by thinking about what's important to the main users of the document. For developers, bookkeeping is crucial and so you might emphasize the identifying numbers for each screen. Stakeholders may need the opposite view, where the supporting information reminds them how one particular screen meets their business goals, and those annotations that tie design decisions to business goals are emphasized. Designers may be more interested in rationale to help them understand why certain elements are present and more important than other elements

What contextual information will you include for the whole document? Most wireframe documents have at least one introductory page that sets the context, giving information like project information, milestones, and version history. Keep in mind that it's difficult for people to flip back and forth between the introductory pages and the interior pages, so don't put anything on the introductory pages that is essential to understanding what's inside. For example, developers may need to know what changed between versions of the wireframe, but version histories are typically included on the introductory pages. If your wireframes are geared toward developers, make sure each page includes its own version history. (Better yet, ask them what kind of information they want on each wireframe.)

Will you render wireframes on paper or in HTML? Here's another good way to get a group of designers worked up: Ask them whether it's better to do wireframes on paper or in HTML. Different teams do it differently, and though you may have a preference, you may find yourself in a situation where you have to go against your instincts.

Boxes and Arrows has several good articles on wireframing, and one in particular about using HTML for wireframes called "HTML Prototypes and Wireframes: All Gain and No Pain." Check this out for more information on using HTML for wireframes.

In the meantime, Table 10.4 a side-by-side comparison to get you started.

TABLE 10.4

Comparing HTML and Paper Wireframes

	HTML Wireframes	Paper Wireframes
Ease of creation	If you're handy with HTML, creating wireframes can be straightforward, but that depends on the complexity of the functionality you're trying to implement. More complex functions require more advanced programming—time perhaps best spent elsewhere.	Although you could go all out and render your wireframes nicely in an elaborate illustration program, this is hardly required. Presentation programs like Microsoft's PowerPoint or Apple's Keynote make putting together a deck of screen illustrations fairly easy.
Maintenance	Though small individual changes can be easy, alterations that affect many screens can be time-consuming.	Some drawing programs like Microsoft Visio and OmniGraffle make it easy to make changes throughout a document by adjusting a master shape or a background image.

TABLE 10.4 *Continued*

Comparing HTML and Paper Wireframes

	HTML Wireframes	Paper Wireframes
Versioning	HTML comments (enclosed in <!-->) make it easy to keep a version history inside the electronic version, but showing how the design evolved may require more HTML mojo than you have. For example, you might put each version in a separate layer and turn layers on and off to compare. HTML editors in general do not include track changes functionality like you might see in Microsoft Word. On the other hand, if your organization uses version-tracking software (like CVS), you can use that in conjunction with your HTML files to track versions.	With paper, the version history can appear adjacent to the wireframe itself.
Saving time and reuse	The jury is out on whether creating wireframes in HTML actually saves any time down the road. In some cases, the HTML needs to be dissected so much that it doesn't actually help the development team.	With paper, there's no time-saving in development, of course, though there may be an opportunity to reuse them for usability testing.
Documenting functionality	No one has yet found a simple way to incorporate annotations into HTML wireframes. Such documentation frequently appears at the bottom of the HTML page, or stuck in a tooltip that only appears with the correct positioning of the mouse.	Paper is ideal for capturing annotations.
Demonstrating functionality	No doubt HTML offers a more realistic demonstration of how the system will behave, allowing stakeholders to click through screen flows, forcing them to envision the system in more than just one dimension. The more realistic experience can support arguments against poorly conceived ideas offered by stakeholders (though even the best evidence may not shake someone from his or her bad notions).	Using paper to demonstrate functionality requires some imagination, a resource not always in abundance in wireframe review meetings. Because paper is so different from an online experience, you may find out too late that your stakeholders came away from the demonstration with a different set of expectations.

What program will you use to create wireframes? For paper wireframes, there are only a handful of dedicated "wireframing" tools on the market, and many people use other applications like OmniGraffle or Microsoft Visio or Microsoft PowerPoint. This decision should be driven by what level of fidelity you want to represent in your wireframes and your level of comfort with the application. Some designers use, for example, Adobe Illustrator, a very powerful, very complex illustration application with a very steep learning curve. There's no point in putting yourself through that ordeal unless you're either already familiar with the tool or are prepared to hike up that tough mountain trail. Even if you are willing, project timelines generally get the upper hand and you may be forced to work with whatever you can get to perform the fastest. For HTML wireframes, any available HTML editor—Macromedia Dreamweaver is popular—should do the trick.

To list the pros and cons of each tool would be futile—opinions about tools vary more than opinions about wireframes. Frankly, pencil and paper is perfectly good if it allows you to accomplish your goals.

Pain-Free[1] Wireframe Creation

OK, so every deliverable involves a little bit of pain, but design documentation—such as wireframes—helps alleviate the potential for much worse pain later in the process. Still, there are a few things you can do to facilitate creating this document.

Make a few lists before you start

Like most deliverables, wireframes require a bit of planning before sitting down to create them. There are a few lists you should make before you start wireframing. First, draw up a list of **scenarios** to identify all the different situations in which people will be using the site you're designing. You can take these scenarios right from the user profiles or use cases. These can be fairly high level, for example, "a user opens a new checking account." The wireframes may need to contrast the experience in two different contexts. On an intranet, for example, you may need to distinguish between administrative users and non-administrative users.

There are a handful of factors that vary across the scenarios, things like the kind of user or the date, or which products are being purchased. Planning out these

[1] Complete lack of pain is not guaranteed by the author or publisher of this book.

specifics is important because it allows you to be consistent in your wireframes. Imagine, for instance, that the wireframes illustrate a shopping process. A product is searched for, selected, customized, put in the cart, and purchased. This process encompasses several different screens. To make sure your stakeholders understand the experience, you should use the same product on all the screens.

This example may be self evident, but dig down a little deeper and you'll identify all kinds of variables that come together to reflect a story—dates, names, locations, search terms, currency amounts, quantities, product types, document types, contact information, identifying information, and so many other things. The more disparity between these variables in the wireframes, the more difficult it will be for readers to follow the experience. Once you identify the variables that are crucial to a user experience, write them down for each scenario so you can remain consistent.

Perhaps most importantly, create a list of **screens** to plan out which wireframes you're creating, for instance, your log-in screen, or your checkout screen, and every other significant screen a user will encounter as he or she clicks through the system. Doing this list in advance can help you decide what to call the screens, how to relate them to the site map or user flow, and will help you plan how much time it will take you to create each wireframe, and therefore the entire document.

There are several factors to consider when building these lists:

- **Importance:** Some screens or scenarios are more important than others because they are crucial milestones in the user experience. A product page on an e-commerce web site or a contact listing on an intranet, for example, are major landmarks in those systems.
- **Timeline:** If you only have a limited amount of time to create the wireframes document, be sure to prioritize the list of screens, estimate the amount of time it will take to create each screen (double that for good measure), and then only create those screens you can within the given timeframe.
- **Complexity:** Some screens will be inherently more complex than others because they have to support complex actions by the user, or they represent complex scenarios.
- **Politics:** Important people may have a lot of political capital invested in particular screens. Though some screens may not be difficult or essential to the user experience, they may be important to the people paying for the project.

Consider the complexity of the system

As applications grow more sophisticated, it will become necessary to distinguish between screens and views. A home page, for example, might appear differently depending on whether the user is logged in or not. A wireframe, especially if presented on paper, is static and will show one particular view, and therefore the scenario must be clear.

You can try to incorporate multiple views in a single wireframe, in order to document how the screen will respond to input from the user, or to highlight the differences between different views. But ultimately, trying to squeeze too much information onto a page could cause more confusion.

Use a simple numbering scheme

Individual wireframes may be identified by a numbering scheme, like the sections of a book.

You can use two numbers separated by a dot, such as X.Y, where X represents the major flow or navigation area and Y represents the page within that flow or area. The home page is always numbered 0.0. The first page of the first main navigation section would be 1.0. This approach works so long as the site can be easily divided into major flows or navigation areas.

You may find that your site is much more abstract—you're creating wireframes of templates, not specific pages in the hierarchy, for instance. In this case, your numbers could refer to template type or page type. So pages all derived from a particular template or matching a particular type of page (like a product details page vs. a product listing page) would be grouped under the same number.

The hardest part with numbering is deciding what to do when the architecture changes dramatically. You may have entire sections removed. Do you renumber? The answer varies depending on how well entrenched the numbering scheme is. On some projects, the screen numbering scheme is used in other documents, like a content inventory or a site map. To avoid having to redo those documents as well, you may continue to use the old numbers, even if they do lose some in the sequence. Regardless of your scheme, every wireframe should have a unique number, even if it represents only a slight variation from another screen.

Always include an introduction

Wireframes without an introduction are like medication without a warning label. Since there's always the risk that you won't be there to present your document,

a one- or two-page introduction will spell out the context by indicating for the reader what's described (and what's *not* described) by the wireframes.

Risks: Don't Get Framed

There's a reason wireframes are controversial, and it's not just that they represent the crossroads of design and architecture. As a complex document that addresses a lot of different aspects of system design, wireframes present lots of risks. Don't get sucked in to the convenience of wireframes without considering these potential pitfalls.

Plan before you build

Once they're familiar with wireframes, project participants may want you to create them as soon as the project starts. At the beginning of the project, remind your stakeholders and team members what role wireframes will play, when they will be delivered, and what you need in order to create them. You may make wireframes look easy, but they don't grow on trees, so your stakeholders should not expect them to.

Establish priorities before doing layout

Making your wireframes look too realistic can convince stakeholders that they are looking at the final product, or an early draft of the final product. If this is what you're going for, great, but in your exuberance to produce a quality deliverable, you may find yourself needing to offer plenty of caveats.

TIP ▸ If you have to offer disclaimers about the look, layout, and design of your wireframes, you should make the wireframes less realistic.

Although it may be tempting to create high-fidelity wireframes, you may spend way too much time thinking about how the screen should look and not enough time thinking about how it should work. Before sitting down to do a wireframe, identify the objectives for the deliverable itself: What messages are you trying to communicate? What ideas are you trying to hammer out? If you start brainstorming in a direction that takes you away from these objectives, you need to be honest with yourself. Be ruthless about what appears in the wireframe and what does not. If you've scheduled your wireframing for the beginning of the design process, you still have many design decisions to make: Do not cheat those decisions by making them too early.

Start with simple labels

Names stick. It's an unfortunate reality. A placeholder hastily slapped on to fill a blank on a wireframe can last through the life of a project, if for no other reason than that everyone else is unwilling to decide on a real name. If these labels make it into the final design, it's either because they went through testing and users liked them, or because the project team is lazy and didn't come up with something beyond a placeholder. The words in the design are as much as part of the design as the color.

Wireframes could include caveats indicating that labels are temporary, but even caveats are not a surefire way to get labels under careful scrutiny. You can also format them to look temporary, using all-capital letters or enclosing them in some kind of brackets. At the very least, use the plainest language possible, in case the temporary labels become forever tattooed on the site. Your best bet is to include an activity in the project plan to review labels.

Design for maintenance

One major advantage to keeping wireframes low-fidelity is maintenance. It can be difficult to keep your wireframes up-to-date with the latest changes. Clients might ask you to update a wireframe with information that's not immediately relevant to it. For example, if the wireframe gives an overview of the content priorities on the screen, the client might ask you to fill in the specific content for the screen. You could probably squeeze the content in, but then the document would become much more complicated to maintain.

On the other hand, even relevant updates may seem like a waste of time depending on where you are in the process. For example, changes to the screen's contents identified during the final design phase—when the graphic designers are creating the actual screen designs—may be pointless to translate to wireframes, since the wireframes are no longer serving as a guide to the design team.

Here are some strategies for keeping wireframes up-to-date:

If the tools permit, tie deliverables together, so that updates to the copy deck, for example, are reflected in the wireframes. If your wireframes are based in HTML, this is theoretically a straightforward exercise. Microsoft Office applications like Excel, Word, and Visio have these capabilities but can be a little temperamental when trying to share them among many people.

If the tools permit, allow the wireframes to draw upon a library of shared elements—parts of the wireframe that are used repeatedly. This

allows you to make changes to an element in one place and have it changed throughout the document. Microsoft Visio permits nested backgrounds, which means that any page in a Visio document can have several backgrounds that it shares with other pages in the document. By establishing a strategy for using these backgrounds early in the process, you can know how to leverage them effectively in your document.

Make changes on the fly. One advantage to showing wireframes on screen is that you can make changes in the document as the discussion progresses. One former colleague used to keep Visio open during requirements meetings, and used it to begin compiling wireframes even before the design phase. This allowed him to respond to changes in requirements quickly and to capture his ideas as they were occurring to him. On the other hand, it also ran the risk of committing to a design idea before fully understanding the scope of the site. A less risky approach is to make changes as they are being discussed in the feedback meeting. This eliminates losing any comments between the client's mouth to your ears to your notebook to your desk to your computer, and allows stakeholders to respond instantaneously.

Don't update wireframes. You may decide that keeping the wireframes up-to-date just isn't worth it. They serve a particular purpose in your process, and once they've outlived that purpose, there's no point in investing further time into them. It's a legitimate decision, to be sure, but one that needs to be weighed very carefully against the disadvantages.

The brute-force approach is simply to build time into the project schedule to keep wireframes up-to-date. Project teams rarely plan for enough time for documentation updates, thinking they won't take long to do. But initial rounds of feedback and the corresponding updates may require as much if not more time than it took to create the first draft. This is perhaps unique to wireframes because they involve such a substantial change in how we perceive the project: Wireframes take us from intangible requirements to somewhat tangible screen concepts.

Use sample content consistently

Wireframes for even the simplest sites represent screens meant to address particular scenarios and particular user goals. To put wireframes together effectively, you may need to identify a sample scenario, ensuring that the wireframes present a coherent experience. In a commerce-enabled web site, for example, the wireframes might show a product screen, a shopping cart screen, and a

checkout screen. If you've included any kind of sample content, the reader of the wireframes will perceive the progression of screens as part of the same scenario. If the sample content on these screens is not consistent, the reader will be confused. The appearance of some other product in the cart, for example, can disrupt the flow of the wireframes. Though this is more of a presentation risk, it must be addressed in the creation of the wireframes. If you've anticipated the scenarios you're illustrating through the wireframes, you can identify the crucial variables in each scenario—dates, names, addresses, products, and other application-specific widgets—and make sure these variables stay the same throughout the wireframes.

TIP ▸ To ensure a consistent flow in the wireframes, plan the crucial variables—dates in a scheduling application, or products and shipping details in a commercial application—in advance.

Presenting Wireframes

If you thought creating wireframes was fraught with risk, wait until you present them to a new client for the first time. Some of the mitigation measures you took to avoid the risks in creating the wireframes will help in presenting them. Whatever you do, though, don't go into a meeting without prepping the meeting participants beforehand.

For new clients who have never seen a wireframe, hold a prep meeting to show them samples, setting their expectations of what they'll get.

For clients who have seen wireframes before, have a conversation with them about what they liked and didn't like about the wireframes they saw in the past. If they ask for greater detail, you have an opportunity to set their expectations about what you'll show them. Bring a sample wireframe from another project—one in a format that you'd like to use for this project—and get their feedback on it.

If you're new to a project team, ask them to show you wireframes they've used before and describe what worked and what didn't work. Ask them about how they incorporated wireframes into their process and if the level of detail was sufficient for subsequent activities.

With expectations set, you may be able to address many of the challenges described in this chapter.

The Meeting

As much as sittir
the advantage to
Run a good mee
a good thing is
go on longer th
only one reason
answering, righ

To keep a tight
the purpose of
answered at the
meetings.

The buy-in m

Suppose you h
is to show it ir
ing how users
the navigatior
narrate the us
themselves in

On the other
or 100 screen
case, the mes
covered ever
marathon m
structure for
series of buy
be more foc
this point, p

For each ki
thing that s

For clients,
better or w

- **Do the**
  or inter
  illustrat

The Meeting Purpose

As much as sitting in meetings may be the bane of your professional existence, the advantage to calling a meeting yourself is that you get to set the agenda. Run a good meeting and you'll become everyone's best friend (whether this is a good thing is your call). Wireframes meetings are difficult to run and may go on longer than planned because of the potential for distraction. But there's only one reason you'd call a meeting, right? There are some questions that need answering, right? That's why you're in the meeting: to answer your questions.

To keep a tight rein on the meeting, always set an agenda and always announce the purpose of the meeting by quickly running through the questions you need answered at the top of the meeting. With wireframes, there are three kinds of meetings.

The buy-in meeting: Selling an idea

Suppose you have a new concept for navigation. The best way to sell the idea is to show it in action. Wireframes can be a good way to do this, demonstrating how users will click from one screen to the next, how the content supports the navigation, and vice versa. In this case, a narrative approach—in which you narrate the user's path through the experience—helps meeting participants put themselves in the user's shoes.

On the other hand, you might need buy-in on the entire system—a deck of 80 or 100 screens that document every nuance of the proposed web site. In this case, the message is comprehensiveness: You want to show that the team has covered every scenario and addressed every requirement. You may be facing a marathon meeting to go through every screen, in which case a scenario-based structure for the meeting is best (see below). If, however, you've conducted a series of buy-in meetings on portions of the site, the final buy-in meeting can be more focused, reminding stakeholders of the process you've taken to get to this point, providing overviews of key scenarios.

For each kind of participant in the meeting, the hook will vary. That is, the thing that sells participants on the idea will depend on who they are.

For clients, the people sponsoring and paying for the project, there are—for better or worse—a couple motivations behind the approval of a concept:

- **Do they get it?** Understanding is crucial, especially for unusual requirements or interaction models that have little precedent. Wireframes can be useful for illustrating an idea, but it's difficult to measure understanding when much of

the conversation is still in the abstract. If you have a comfortable rapport with your clients, you might ask them to walk you through the wireframes after you've explained it once to see whether they really got it.

- **Does the idea suit their ego?** Everyone wants to believe that their clients are ego-free. (And if you are on the client side, you are *clearly* an exception.) On the other hand, if a concept incorporates one of their ideas or somehow addresses something that they've previously raised, this will go a long way toward getting their buy-in.
- **Does the idea make sense to their understanding of their business?** The "their understanding" part is crucial to that question. You may have certain notions about how your clients' business should operate, but these are useless unless you've convinced them that such an approach is appropriate. Either way, the wireframes must show that you've addressed a crucial part of their business. For many stakeholders, the system you're building (especially public-facing web sites) is incidental to the business—an additional sales channel that doesn't compete with their more traditional channels, for example. In the government, making information available on the web is a legislated mandate, but there are lots of mandates with which agencies comply that aren't high priorities for them.

Though these questions are mostly about stakeholders, they are relevant to the other meeting participants, designers, and developers. To a certain extent, understanding, ego, and business perspective play a role in their reaction to a concept. There's one more question that you need to consider when presenting an initial idea to other members of the team:

- **Can they build it?** Designers and developers will be looking at wireframes imagining the effort it will take to translate the design into something "real." The best way to prepare for this is to first ask yourself all the questions you think they'll ask you, and then have an informal conversation with these project participants—running the ideas past them in an informal setting allows you to anticipate the most difficult question of all: Is this realistic?

Ultimately, the best way to get buy-in on an idea is to have everyone participate in coming up with it. Project team members who contribute to an idea will be supportive when selling it to stakeholders. Stakeholders who contribute to an idea will be more likely to (and let's be honest here) fund it.

The feedback meeting: Getting input

If a buy-in meeting is to sell a mostly complete idea, a feedback meeting is to flesh out a somewhat complete idea. Because the conversation will be much

more elaborate—discussing the ins and outs of your proposed solution—the scope of the meeting must be shorter. You can't get feedback on a deck of 80 to 100 screens in one sitting (unless your team likes sitting for ten hours at a time).

For the feedback meeting, the agenda should focus on exactly what you need feedback on. Even if there are many other things to work out, prioritize your questions to keep up the momentum of your design process.

Feedback isn't quite brainstorming. Brainstorming starts with a clean slate and permits limitless ideas, more or less. The focus of a feedback meeting is less "how do we solve this problem?" and more "does this approach effectively solve the problem?"

When presenting wireframes in this context, therefore, you'll need to show enough of the concept to explain it, but not so much as to stifle the feedback. If a solution seems mostly complete, participants may wonder what they're doing there and realize that "feedback" was a euphemism for "buy-in." Use feedback meetings to answer questions you may have about the approach you're taking.

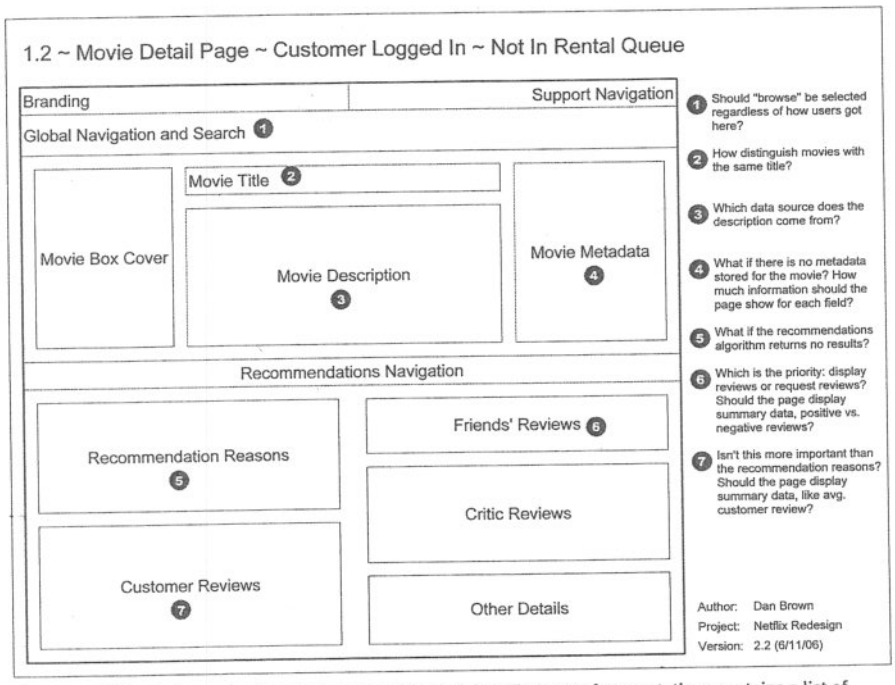


Figure 10.8 This wireframe is all ready for a feedback meeting. The space for annotations contains a list of questions for the meeting participants to consider. By putting the questions right on the wireframes, the meeting agenda is built into the document.

The brainstorm meeting: Developing wireframes together

When you brainstorm around wireframes, you need to keep the scope in check. Collaboratively designing every element on every screen will drive the project team crazy. To control the scope of brainstorming meetings, focus them around a particular scenario, a series of screens for supporting particular user behaviors, or even around a single type of user. For example, one meeting might trace a user's path as he or she tries to buy a particular product online, while another might trace a customer's attempt to find out when the package will arrive.

Once you have a focus for the meeting, don't try to dissect every aspect of the screen. Instead keep the agenda tight by addressing only the salient parts of the screen contents—the information users need in a particular scenario, the functionality that will move them toward their goal, or the messaging priorities for different entry points. Straightforward, tangible, and practical goals make for productive meetings.

How to Structure Your Wireframes Meeting

There are several different ways to structure any kind of meeting about wireframes. Often, the narrative approach is easiest because it follows the path of the user, but you may have a situation where that approach is inappropriate—if there are many different ways to walk through the site, for example, or if the screens are very abstract. Ultimately, the choice of structure is driven by the purpose of the meeting: Use the questions you need answered to define a way to present the wireframes.

The narrative approach

Wireframes can be very confusing to other people on the project team. Very abstract wireframes don't look like screens and participants may not understand what they're looking at. By contrast, very detailed screens may cause tangents in the presentation because they offer so much to respond to. A narrative-style meeting, based on a series of scenarios, can establish context and guide the conversation.

In selecting scenarios for wireframe presentations, start with the basic processes—with no errors, exceptions, or deviations. For example, on a banking web site, describe the process for setting up a basic checking account—a process where

the user doesn't already have another account, or where they don't get a credit card to go along with their new checking account, just the basics. Depending on the purpose of your meeting, you may not be able to get through much beyond that, especially if the purpose of the meeting is to get feedback or brainstorm the details of the process.

Once you have run through the basic process and you're confident that the other meeting participants get it, you can show how the process changes in different scenarios.

There are two ways to structure meetings using a narrative approach: by flow and by user, and frequently a combination of both. Each "chapter" in your meeting could cover a different flow, a different discrete task performed by the users, for instance, finding a product, browsing the online store, committing a product to the shopping cart, checking out. This approach makes sense when every user is going to have pretty much the same experience.

On the other hand, the web site experience may be different for different users—a content management system where different users perform different tasks, for example, or a banking site where the experience varies depending on the kinds of accounts a customer has. In these cases, your meeting structure is a little more complex. Some "chapters" of the meeting are processes that all users have in common—getting help or logging in, for example. The rest of the meeting should then be structured based on different user types.

The "content type" approach

If your site contains various levels of information hierarchy and several different screen designs on each level, consider presenting each level in its entirety. For example, if a storefront has three different kinds of category pages and four different kinds of product pages, this approach suggests you present all three category pages followed by all four product pages. (In the narrative style, for comparison's sake, you would present a category page and then the product page that follows.)

Although the exact style of presentation will vary with the purpose of the meeting, the best way to use this approach is to present all the screens of a particular type and then analyze or critique them all together. The structure of each screen of a given type may vary, but the overall purpose of each screen is the same (otherwise they wouldn't be of the same type) and some feedback may

apply across the board. Be sure to allow participants to see all the wireframes at the same time (presuming there are no more than five or six variations on a screen).

The "priority order" approach

During the design process, the team may identify the most important screens in the system. These screens generally represent the main reason why someone has come to the site. On a web-based storefront, it could be the product page, the screen that makes or breaks the customer's decision to buy something. These are the 20 percent of your screens that will take 80 percent of the traffic. It's the screen where most of the action happens (This used to be the home page, but these days, with the proliferation of search engines like Yahoo! and Google, many customers bypass the home page entirely, so its value is rapidly diminishing.)

One way to conduct these meetings is to start with that screen. This approach follows software engineering best practices by addressing the "highest risk" items first, from which follows the idea that by solving problems on that page, you address global problems throughout the system.

By dealing with these key screens first, you avoid the risk of saving them until later in the meeting when important people might have left, or the participants have started to tire. Missing someone's input on crucial screens can cause further delays down the line.

This is not to say that the paths users take to these important screens are insignificant. There's a hierarchy of pages below that most important screen, those that allow customers to take the next step and those that bring the customer to that screen. Rank these pages according to risk—which ones have the greatest impact on the project—and present them in that order. The screens can be introduced relative to the most important one, for example: "This is the page users see immediately following the project page once they've added the product to their cart." You may not be able get through more than two screens, but you've at least reviewed the two most important screens and gotten implicit feedback on the rest.

The risk with this approach is that the team sees screens out of context—not necessarily how the customer will see them—and it may be difficult to imagine the customer experience.

Risks: Don' t Ge

Getting bogged dc

Regardless of your n
the relative prioritie
sample content—or
sation focused by av

If you can't avoid p
couple things you c
you. Most importa
frames, it's easy to
ing the purpose of
and what you want
on topic. That bei
help mitigate this
talk about it.

When the conversa
errors, you need tc
ously, but that you
issues publicly—o
meeting minutes
you've heard their
conversations abo
tent errors do not
sion of the docu

Steer the meeti

Like content, the
versation at hand
keep the wirefra
address later.

Discussions abou
concerns. Try to
or priority.

Risks: Don' t Get Your Wires Crossed

Getting bogged down in content

Regardless of your meeting's purpose, the focus of the conversation must be on the relative priorities of the elements on the screen. Wireframes without any sample content—or with unrecognizable sample content—help keep the conversation focused by avoiding the issue entirely.

If you can't avoid putting sample content in your wireframes, there are a couple things you can do to prevent the meeting from running away from you. Most importantly, set the agenda and the expected outcome. With wireframes, it's easy to dive right in and start talking about screens. By establishing the purpose of the meeting up front, listing the wireframes you'll discuss and what you want to talk about for each one, you have a means of staying on topic. That being said, making your wireframes as accurate as possible can help mitigate this risk: If the content is accurate, they don't have a reason to talk about it.

When the conversation does get into label names, editorial issues, and factual errors, you need to reassure your stakeholders that you take their concerns seriously, but that you need to address other issues at the moment. Capture their issues publicly—on a whiteboard or flip chart—and include these issues in the meeting minutes or an informal follow-up email. This will make it clear that you've heard their concerns and give you a means for moving past extended conversations about content. The ball is now in your court to ensure these content errors do not make their way into the final product—or even the next version of the document.

Steer the meeting toward priorities

Like content, the design of a wireframe can distract stakeholders from the conversation at hand. Many of the mitigation strategies are the same: set an agenda, keep the wireframe as devoid of design as possible, and capture design issues to address later.

Discussions about design, however, can by symptomatic of deeper structural concerns. Try to turn design conversations into conversations about structure or priority.

TABLE 10.5

Redirecting Wireframe Conversations

They Say	You Say
I don't like the color of the button	The color makes the button stand out. Do you think that the button should not stand out so much?
I don't like having to scroll to see this information	The screen has important information escalated to the top and we've tried to avoid diluting the important information with information that's not as important. Is this information more important than the stuff above the fold?
This isn't our corporate typeface	The wireframe won't accurately reflect your corporate branding, but the final design will. Do you think this information needs to be more visually prominent than other things on the page?
There is too much going on; it's too busy	Of all the information on the screen, which is the most important? Let's rank the different pieces of information on this screen.

Unlike conversations about content, which usually have no bearing on the underlying structure, comments about design can reveal important requirements about the task at hand.

Keep an open mind

Requirements are the bread and butter of software engineering, and by extension, web development. A requirement is a statement describing what the system is supposed to do. They usually come in packs (read: enormous documents that no one ever reads more than once, or past the first few pages). Good requirements are stated in such a way that they don't dictate design. For example, "the system shall allow the user to remove an item from the list of items they would like to purchase." It is usually through requirements that you gain an understanding of what you're supposed to design.

The wireframes are a response to requirements: Through the requirements you learn what the system is supposed to do, and the wireframes show how the system does it. The dichotomy is rarely so clean, but this is the theory anyway.

The risk is that in running through the wireframes, some stakeholder will have a sudden epiphany. If she's nice, she'll admit that she's proposing a new feature. If she's not so nice, it'll sound something like this: "Why doesn't the page show product recommendations based on previous purchases like we talked about at the kickoff meeting? I distinctly remember describing it in great detail."

On rare occasions, wireframes are used specifically for this purpose—putting concepts in front of stakeholders or customers to get them to think through everything the system should do. Mostly, however, wireframes are used after the requirements have been defined and approved. Still, it's generally inevitable that a new requirement will appear over the course of reviewing the wireframes.

Part of this is human nature. Until you see something, it's hard to think through all of its implications. From a document preparation point of view, there is little to be done about that—this is a project management and methodology issue.

Still, there is at least one thing you can do to avoid being surprised by important requirements: Create quick-and-dirty wireframes before the requirements process is over. These wireframes are only for you and your most trusted compatriots, a tool to validate the requirements and help surface any that you might have missed. Focus not on meeting user needs but on addressing all the requirements. The aim here isn't to develop award-winning design, but to poke holes in the requirements document. If you do a quick version of the design process and find unanswered questions, you've found a hole.

Otherwise, the ways to mitigate this risk include better project planning, better expectation setting, and better presentation methods.

Manage team interactions

Perhaps this isn't a risk anymore, but in the early days of web design, wireframes were controversial because they crossed the line between information architecture, interaction design, and graphic design. (That's one of the reasons this chapter advocates leaving layout and design out of wireframes.) As a potential territory issue, wireframes can create a lot of stress on design teams.

Other than keeping wireframes focused on priorities, there is little you can do in the document itself to mitigate this risk—it's more of a project management and methodology issue. Developing wireframes collaboratively, however, offers a key advantage to mitigating the risk: By working on them together, the team

creates joint ownership in the design. With joint ownership come fewer territory issues.

As described above, brainstorming is a tricky endeavor, requiring careful orchestration: It's very easy to go way out of scope or way off topic. The bottom line is that if problematic team dynamics are a real risk in your situation, brainstorm wireframes together, but do your situation analysis beforehand and take the role of facilitator.

Wireframes in Context

Wireframes can't exist by themselves. They grow out of work and documentation, and they're just one stepping-stone among many along the way to the final product. This section talks about the bigger picture: how wireframes relate to other documentation and to the project as a whole.

Relationship to Other Deliverables

As powerful as wireframes are in representing many aspects of the user experience, they are not self-contained documentation. They rely on an understanding of user needs and an overall strategy for the foundation. They cannot describe 100 percent of the user experience, and they must work with other design documents to provide a complete picture.

Wireframes and user-needs documentation

In the deliverables that describe users—personas and usability testing documentation—you have expressed, in essence, checklists. Your solution will work if it successfully addresses everything specified in these checklists. The wireframes represent one view of that solution, but they may not address everything expressed in the user-needs documentation.

Whether it happens in the document itself or in your presentation of the document, you must show how the wireframes address previously defined requirements.

At the same time, because wireframes represent only part of a solution, it would be impossible for them to address every requirement. To show that you aren't neglecting these requirements, but merely waiting for a more appropriate time to address them, you can explicitly call attention to them. For example, your

usability rep
in a casual t
leaving cont
the requiren
clear in the
or personas
ment can als
sented to hi

Requirement 1.1
users to view mo
movie to their rer
Requirement 1.1
the movie title, a
movie, and the n
director.
Requirement 1.1
users to add the
Requirement 1.1
whether the mov
Requirement 1.1
reviews written b
most popular rev
and the most rec
Requirement 1.2
users to rate mo
reviews.
Requirement 1.2
users to easily a
and five stars) to
Requirement 1.2
the user's movie
recommended n

Figure 10.9 This
wireframes addr
kinds of docume

Wireframe

This book c
foundation
the competi
practices or
marizing hc
comparison

Another str
ture for the
wireframes

usability report might show that users responded positively to content written in a casual tone and with lists rather than long paragraphs. Of course, if you're leaving content out of the wireframe, you cannot show that you are addressing the requirement; this needs to happen elsewhere. To make these distinctions clear in the wireframes document, you can include references to usability results or personas on individual screens to show what you've addressed. The document can also include a table at the beginning that compares the screens presented to high-level goals or requirements.

	Flow Chart	Wireframes	Tech Spec	Final Design
Requirement 1.1: The system shall allow users to view movie information and add a movie to their rental queue.	●	●	●	●
Requirement 1.1.1: The system shall display the movie title, a short description of the movie, and the movie's main actors and director.	○	●	◑	●
Requirement 1.1.2: The system shall allow users to add the movie to their rental queue.	●	●	◑	●
Requirement 1.1.3: The system shall indicate whether the movie is in their queue or not.	○	◑	○	●
Requirement 1.1.4: The system shall display reviews written by other users, showing the most popular reviews (determined by votes) and the most recent reviews.	○	◑	◑	●
Requirement 1.2: The system shall allow users to rate movies and write their own reviews.	○	◑	◑	◑
Requirement 1.2.1: The system shall allow users to easily assign a rating (between one and five stars) to a movie.	○	●	◑	●
Requirement 1.2.2: The system shall employ the user's movie ratings to determine recommended movies.	●	○	●	○

Figure 10.9 This table shows which requirements are addressed by the wireframes and the extent to which the wireframes address each requirement. The table compares wireframes in the context of requirements to other kinds of documents, anticipating that stakeholders might expect to see every detail specified in the wireframes.

Wireframes and strategy documentation

This book describes three kinds of strategy documents, all of which establish a foundation for building the overall user experience. One of these documents, the competitive analysis, compares your web site with others to identify best practices or differences in what's offered. Wireframes are a useful tool for summarizing how each site presents information and allows an apples-to-apples comparison of screen layouts.

Another strategy document, the concept model, describes an underlying structure for the system. Your wireframes' introductory pages can explain how the wireframes support the underlying structure or organize the wireframes in

terms of the concept model. A concept model represents an overall organizing principle, and so may be concealed from the user in the final product. Your wireframe should acknowledge how it makes use of the model without burdening the users with knowing what the model is.

The last kind of strategy document, the content model, helps plan content migration and creation. Content models identify all the content contained on the site while wireframes describe how the content will be displayed. If the wireframes represent templates, the content model should reference the wireframes to show what kinds of content will be presented in what templates.

Ultimately, the goal is to show the stakeholders that these preliminary deliverables have built a foundation for later work. Although this may feel like a ploy to justify the work, the real purpose is more important: to keep the stakeholders focused. The strategy at the beginning identifies what's important to the project, and the design realizes those priorities.

Wireframes and other design documents

The key to successful design documentation is recognizing that no one document can capture every detail about the user experience. Therefore, there's no need to burden wireframes with information that other documents do better. It's easy enough to show how the different documents relate to each other: A screen in the flow chart is the same as a screen in the wireframes is the same as a final screen design. Use the same reference numbers throughout.

The Impact of Wireframes

Throughout this chapter, I've been very strict about the purpose of wireframes, and most situations call for being a stickler about that sort of thing. As a document, wireframes can be incredibly powerful in communicating many aspects of the user experience—how information is prioritized on the screen, how users interact with that information, how design is derived from requirements and business goals. To paraphrase a superhero, with great power comes great risk. Until your team and stakeholders find a comfortable working relationship and agree on the role of wireframes in the design process, these can be murky waters.

In some design methodologies, wireframes can both represent design ideas and help stakeholders zero in on what they need. Wireframes can be especially powerful for identifying these parameters—or "eliciting requirements"—because

people tend to recognize what works only when they see it. Wireframes, therefore, are an ideal tool for a looser design process that permits developing a design and revisiting it throughout the life of the project. Developers call this an "iterative" approach. Moving right to wireframes after identifying high-level requirements can accelerate the entire process.

Many recent innovations in software development methodologies eliminate wireframes, moving straight from lightweight requirements to functional prototypes (working models). Although this approach offers clear advantages of speed, it requires a certain level of agility from both the project team and the stakeholders. Many organizations may not be able to muster the speed required.

You might wonder whether there's a better way to do wireframes, one that isn't fraught with such risk. But for all their risks, wireframes can be an incredibly useful tool for visualizing the behavior of the system in relatively short order. By seeing and almost touching the solution, team members can identify potential risks with the entire project early in the lifecycle. Removing wireframes from the process entirely burdens other documents with the responsibility to respond to the requirements.

Wireframes in a changing landscape

Rapid changes in technological capabilities and development methodologies won't render wireframes extinct, but will force them to evolve to meet the demands of a new landscape.

As mentioned earlier, there's a growing trend in software development methodologies to move faster, work on smaller chunks of a project at a time, and revise often. Wireframes, as a deliverable with many pages and only a handful of versions, won't survive long in that environment. New tools will emerge that render the original purpose of wireframes—to visualize the solution—obsolete. These tools will allow teams to create functional prototypes quickly. Wireframes will become a tool to provide guidance for the prototype, further compressing timeframes for creating the wireframes.

At the same time, interfaces are becoming more dynamic, especially on the web. It won't do to divide a system into discrete screens because users will be able to do so much on a single screen. Future visions of the Internet focus on a "participation economy," an environment where users contribute and manipulate content more than they consume it. Information priorities, the bread and butter of a wireframe, take on new meaning when systems are designed as containers of contributed (and therefore somewhat unpredictable) content, rather

than portals to a centralized content source. The "old" web is one of back-and-forth communications: You send me your credit card information and I'll ship you some books. The online bookstore, in that model, was a central content source for books. In the participation economy, information about books is distributed—anyone who reads (or sometimes doesn't read) a book has something to say about it—information about customer needs is distributed, and even fulfillment is distributed. Navigation is difficult to plan when you're not exactly sure what's being navigated. And, of course, describing content becomes almost completely meaningless when your stakeholders' customers are writing it. The wireframe as a representation of information priorities and content structure may be no longer necessary.

But the wireframe itself won't go away. Humans will always gravitate toward pictures and toward simple representations of complex ideas, which are—in the end—the whole point of a wireframe.