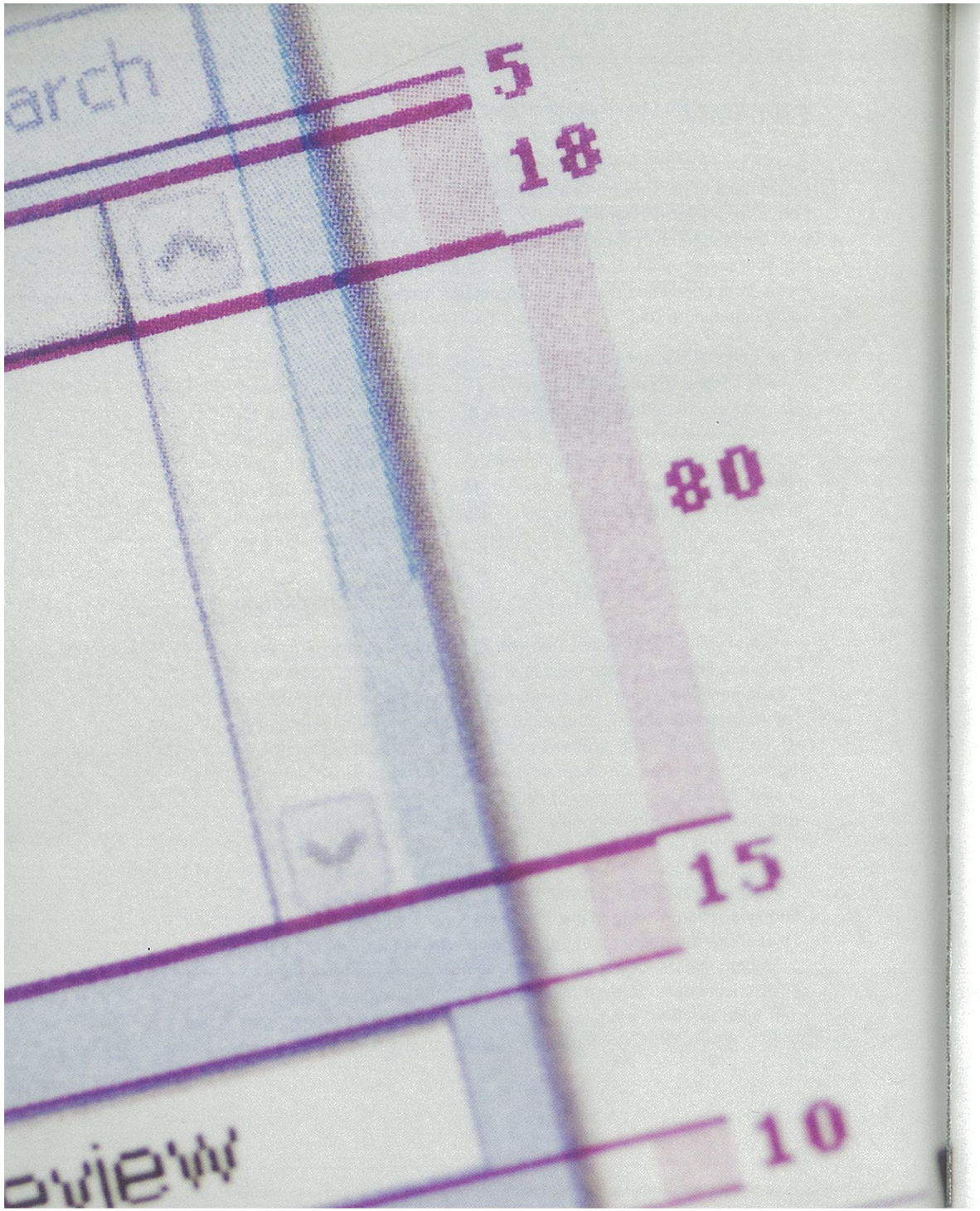


CHAPTER 24

Communicating Detailed Design

The fate of your design is determined in part by how effectively you communicate about it with both stakeholders and engineers. As discussed in Chapter 22, frequent collaboration is essential to communicating and evolving detailed design. However, more formal communication is nearly always required. A detailed representation of the final product helps stakeholders be patient during the long implementation process. A clear and detailed blueprint for construction—manufacturing in the case of physical products and production coding in the case of software—helps ensure effective translation from design to finished product. Your communication must be particularly thorough and unambiguous in the event that you're not around during implementation, which may happen if you're a consultant or using outsourced engineering.

A form and behavior specification—abbreviated as the "F&BS" at Cooper, tongue firmly in cheek—describes every visible aspect of the product's form and behavior. This isn't limited to physical controls and pixels; it also includes everything from workflow to evident business rules, such as how long messages stay in a phone's inbox or how much battery time remains when a device

What the specification does not include is any calculation or structure affecting the back end: how server loads are balanced, how the device assembly line will work, or what database tables are involved in a transaction. These important issues are usually beyond a designer's expertise; design engineers and business analysts fill in these last gaps between design and construction.

For most software-only products, the F&BS tells a skilled engineer nearly everything he needs to know. The F&BS doesn't necessarily stand alone, however. It should ideally be supplemented with production-ready digital assets. In some cases, animations of any subtle behaviors help engineers understand the behavior you envision. Software engineers and business analysts might also follow the F&BS with a technical or "functional" specification that describes how an application needs to be constructed.

For physical products, you will of course need a CAD database for manufacturing. An appearance model—a close physical approximation of the product's finished look—will help designers and

't forget that
r spec also
an invisible
lience: people
o will need to
lerstand your
ject in the
ure.

A prototype that simulates the function, fit, and assembly of components can provide a final sanity check before manufacturing.

By the time you develop the text, drawings, and models that make up the specification, the cost/benefit trade-offs have (hopefully) been made. However, it's still important to do a live presentation—or several, if you've chunked your design delivery—for the entire group of stakeholders. This ensures that they at least see the detailed design (even if they never crack open the spec) and publicly agree that the design is ready to go. Moreover, a presentation serves as a public celebration of what might have been a lengthy process. It's difficult to build momentum for design in many companies, so don't pass up an opportunity to show off your work in its best light.

The Form and Behavior Specification

With the current popularity of agile software engineering approaches, many designers are being asked to provide "light" documentation, such as sketches with a few notes. This can work well if you have an established visual system, a relatively uncomplicated product, and a small engineering team working closely with you. However, large or distributed engineering teams can't all collaborate closely with you, and the more engineers there are on a project, the greater the likelihood of inconsistency in skills, judgment, and interpretation of loosely defined specs. Less-skilled engineers are likely to take shortcuts based on what's easier to code if there's any ambiguity in the spec. Any time you don't have a very close relationship with the engineers, such as when your company is outsourcing development, specificity is essential.

Documentation is also an important design tool for ensuring the correctness and thoroughness of your design; skipping it simply means you'll discover design flaws after writing code, which is far more expensive than writing a few pages of documentation. Detailed specs are also indispensable for QA and user documentation. Instead of skipping the detail, designers working with agile teams should provide thorough documentation in an incremental fashion.

Effective documentation combines text and images to describe the anatomy and physiology of a product. If you supply only drawings with minimal text, engineers have to guess at behavior. If you supply only an animated or clickable prototype, they have to deconstruct its form and behavior for themselves. If you supply only text, wireframes, or UML diagrams, the potential for ambiguity is enormous.

You can develop the F&BS as a static document using a tool such as Adobe FrameMaker or InDesign, or you can assemble it in interactive form as an intranet site, which will let you build in clickable cross-references and links to asset files such as icons. Electronic documents also ensure that everyone is looking at the latest version. However, many people seem to prefer more traditional printed documents such as the one in Figure 24.1. Perhaps this is partly because large documents are still easier to read in print than on a screen. Engineers can annotate printed documents and leave them open on a desk without having to switch windows. Also, there's something about the *thud* of a big document on a table that makes people feel it's authoritative. Whether you decide it's more important to save trees or to save your engineers' eyes, the following discussion is equally relevant.

Figure 24.1. A thorough form and behavior spec can be a massive document.

The primary target audiences for the specification are the engineers, quality assurance team, and any business analysts responsible for helping

stakeholders, aside from perhaps a product manager or subject matter expert, don't need to know the design quite so thoroughly. A formal presentation and a casual flip through the document may be sufficient. However, don't forget that your spec also has an invisible audience: people who will need to understand your project in the future, such as new designers, engineers, or managers who may join the team a year after you're gone. I've also had clients tell me the F&BS is an excellent recruiting tool for programmers, since it not only shows they'll get to work on interesting things, but also conveys that this is a company that has its act together.

As with formal communication on the earlier stages of your project, decide on your narrative structure as a team. However, the same contents and structure work well for most situations. A typical F&BS starts with sections covering:

- Background
- Executive summary
- Personas
- A product or service overview

Sometimes the majority of the user and domain analysis document is included as an appendix. Next, the document covers these topics for each primary persona's unique interface:

- An interaction framework overview, including any hardware interaction
- A small set of scenarios representing the key paths
- Form and behavior details for each screen and physical control

It would be tedious and inefficient to document the behavior of every control on every screen; instead, document any widgets or larger components that are shared across multiple screens (or between the screen and physical surfaces) all in one place:

This document may be the only one a reader ever sees, so you can't assume familiarity with the project and your work product to date.

- Common components and interactions
- Visual system overview
- Color palette and CMF specifications for hardware
- Type specifications
- Icons
- The grid
- Screen specs with measurements (may also go in behavior section)

If you have time, document some key principles and rules for expanding upon the design; hopefully the engineers aren't doing this on their own design, but you or another designer may find these useful later.

Background

The main background points to cover are:

- Project mandate, timeline, and overall approach
- Any conventions used in the document

This document may be the only one a reader ever sees, so you can't assume familiarity with the project and your work product to date. Briefly review your mandate, timeline, and approach to the project. Explicitly call out the engineering timeline and any aspect of the product or service that wasn't in scope—this helps readers who are late to the party understand the project context, without which they might judge your work by the wrong standards. Happily, you've probably written most of this already if you did a user and domain analysis document (see Chapter 13). You'll just need to update the timeline and add any conventions used for documenting design, such as, "**Bold text** represents the name of a button or other on-screen control."

Executive summary

Executive summary main points include:

- Quick product overview
- Highlights of big ideas and unusual design components
- Explanation of the design's value

An executive summary is not a must, but is a good idea if you have time because it gives busy executives (and anyone reading the document a few years down the road) a sense of what it contains and whether it's worth reading further. Keep the summary short; explaining the design's key points and benefits in a page is good rhetorical practice anyway.

Personas and critical requirements

Main points for the personas and critical requirements are:

- What personas are and why they're useful
- Overview of the set
- Review of details about key personas
- Review of key requirements agreed upon for each persona

Yes, that's right—you're not done with the personas yet. An effective spec weaves the personas throughout and uses them to explain design rationale, so it's helpful for readers if you reintroduce them. Also, chances are that not everyone reading the document was involved with the design team during the user and domain analysis discussion. Briefly describe what personas are, how they're derived from behavior patterns in the research, and why they're useful. You can probably borrow this section from your user and domain analysis, too. Rather than describing a large set of personas all at once, you could simply show a matrix summarizing the whole set, then introduce details for only the relevant personas as you begin describing each interface.

Product or service overview

The main points of the product or service overview include:

- Any unusual ideas that are the basis for the design
- Experience attributes
- Overview of the form factor(s), if applicable
- For a multi-interface product, overview of who

It may seem odd to describe the big idea behind the product or service in a spec, since presumably everyone knows this by now. However, new people get hired, project funding gets postponed for three months, and people need to be grounded in the project's fundamentals all over again.

Readers will better understand the design language and some of the more subtle aspects of behavior if you remind them of the experience attributes (see Chapter 12). Those little animated touches or other nuances may be a pain to code, but they exist for a reason.

If there's a single hardware form factor shared by multiple interfaces, review it here; otherwise, do so in the framework overview for each interface. Outline the number of distinct interfaces and the personas associated with each. You should be able to borrow the basic outline of this content from your framework presentation.

Interaction framework overview

The interaction framework overview consists of:

- An introduction of relevant persona descriptions, if not already covered
- Overview of the form factor and physical behaviors, if applicable
- Number and functions of key screens
- Any major elements (such as an organizer pane and toolbar) common to key screens
- Navigation and relationships among common elements
- Data objects relevant to the interface

This section is combined with the product overview for a single-interface product. You'll have multiple interface overviews for a multi-interface product. The chapter on each interface should outline what screens exist in the interface and what common structures (such as those in Figure

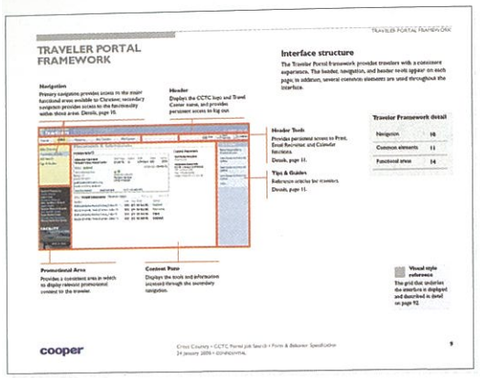


Figure 24.2. A typical framework overview describes the number of screens and the key components.

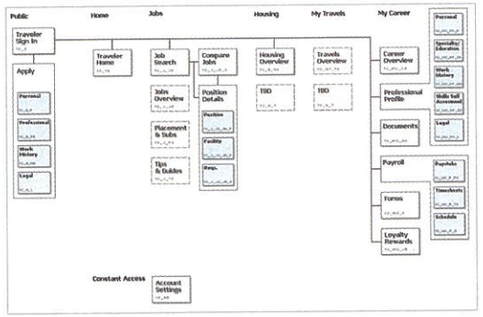
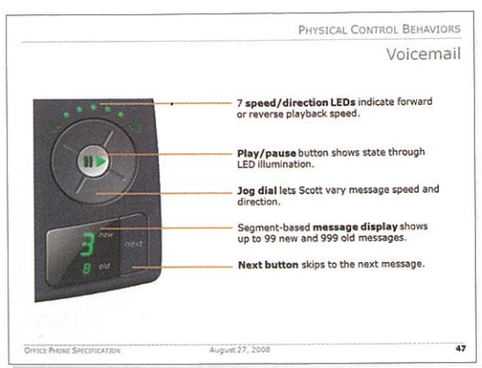


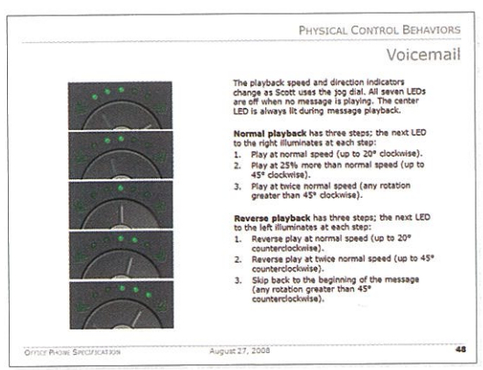
Figure 24.3. A navigation map for an application with many screens.

product follows a hub-and-spoke pattern or has a large number of screens, such as on many Web sites, this is a good place for a navigation map (see Figure 24.3).

The overview provides context for the design details, introduces any new vocabulary, and helps readers stay oriented. Engineers also find it helpful to understand the overall intent of the product design even if they're only focused on a subset of functionality. When there's hardware involved, provide a description of each hardware control, its functions, and its relationship to screen contents, as in Figure 24.4.

If you think this sounds like a recapitulation of your framework presentation, you're right. This is important both because your framework may have shifted since the earlier presentation and because people tend to look at the most recent piece of documentation, forgetting that others exist. For example, a couple of years ago, my team did an abbreviated detailed design phase for just one of several interfaces in our framework design. Due to the tight timeframe, we didn't spend any documentation time on the framework. A year or so later, the client's team was trying to use that form and behavior spec as a basis for the other interfaces, but they kept feeling the need to invent new things that deviated from the framework and added a lot of complexity. When they called me in to help them get back on track, it turned out that the information they needed was all in the framework presentation, but they had stopped using that as a reference tool once they had the F&BS.





Scenarios for each interface

The scenarios section of your document should include:

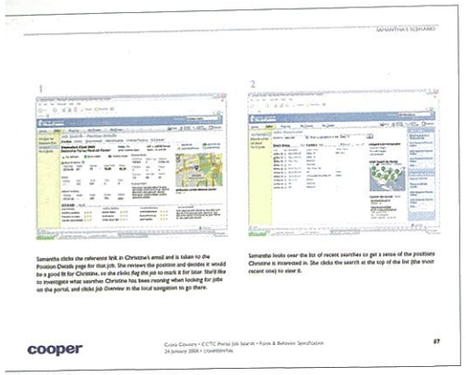
- An introduction of scenarios and their role in the design process
- A list of the scenarios covered
- The individual storyboarded scenarios

As discussed in Chapter 19, scenarios demonstrate the value and intent of the design in a way that no amount of functional description can. The fact that your design is essentially finished doesn't mean you can stop selling it; on the contrary, you must continue to promote its value even throughout implementation, when some unfortunate shortcuts can happen if stakeholders and engineers forget why they agreed to build something difficult.

The fact that your design is essentially finished doesn't mean you can stop selling it; you must continue to promote its value even throughout implementation.

Briefly introduce the concept of scenarios (since you may have new readers) and explain that they focus on key elements of the interaction, so they're not meant to be exhaustive. Don't try to illustrate every possible interaction. Instead, turn your key path scenarios into communication scenarios that show the product at its best. The scenarios aren't there as much to provide specific construction guidance as they are to help readers understand and value the design. For a typical desktop platform, a layout with two screens to a page, as shown in Figure 24.5, allows for details to be visible.

Important as the scenarios are, if you're truly pressed for time, err on the side of documenting behavior detail rather than doing a lot of scenarios; the necessary screen state changes are time-consuming to draw. Don't feel like you have to illustrate every behavior with a scenario. Keep the scenarios simple to make the story more effective.



igure 24.5. An example of scenario documentation.

)verview and details for each screen or unction

'he overview and details section for each screen ır function should address:

- An overview of the screen's or function's purpose
- Screen anatomy
- Contents of individual widgets and data displays
- Behavioral details for any widgets or data displays unique to the screen
- Exact measurements for screen layout (if not in visual style guide section)

'hese elements are detailed in the following ;ections.

)VERVIEW

Give each screen within the interface a name, such as Calendar, Contact List, and Mail. Again, keep readers oriented by providing some context about what each screen is and why it exists. Using the personas in the overview serves as a reminder that everything you've designed has a rationale behind

screen, such as a day, week, or month view in a calendar, identify what they are and why they exist.

FULL SCREEN ANATOMY AND RELATIONSHIPS

Next, introduce the screen's contents. A nearly full-page image with callouts is usually an effective way to do this. On the side of the interface closest to each unique element, list the element's name and a very brief description of what it does, such as:

> **View control**. Switches the calendar among daily, weekly, and monthly views.

Use a line to point to each element. Callout lines look neater if they are exactly vertical or horizontal, with perhaps a few at 45° angles if necessary. Callout lines that overlap each other or at random angles can be confusing, especially with a dense screen.

For particularly dense screens, you may not be able to call out every control or data display. Instead, focus on naming the areas of the screen, such as an organizer and workspace, along with the most important elements within each area. You can then zoom in on each screen area in subsequent pages with callouts identifying every component of a dense area, as shown in Figure 24.6.

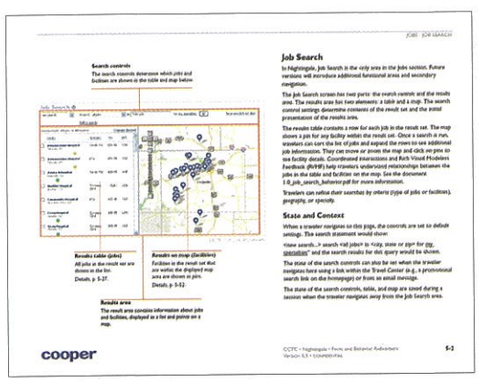


Figure 24.6. The Cross Country job search anatomy page with callouts

If there are any unique relationships among areas of the screen—such as one pane that drives another—describe those here if you haven't already done so in the framework overview.

INDIVIDUAL COMPONENT OR FUNCTION DESCRIPTIONS

Imagine your document as a map that's zooming in from 30,000 feet to 20,000, 10,000, and eventually ground level. If your screen is divided into, say, three major areas (such as an organizer, workspace, and toolbar), you'll want to zoom in on these, then break them down into discussions of every widget and piece of data they contain.

Again, use an image with callouts followed by detailed description, as shown in Figure 24.7. You'll want to do the same for functions and idioms shared across screens. You may also want to describe all of the valid states, possible values, field-level validation, measurements, icon file names, and so forth. See the description of selection behavior and the error state diagram in Figure 24.8 for examples of important details.

However, global controls (or idioms) shared by every screen, such as a toolbar, are usually best explained at the end of the interaction overview, before you dive into individual screens. You may not need to describe fundamental behaviors for

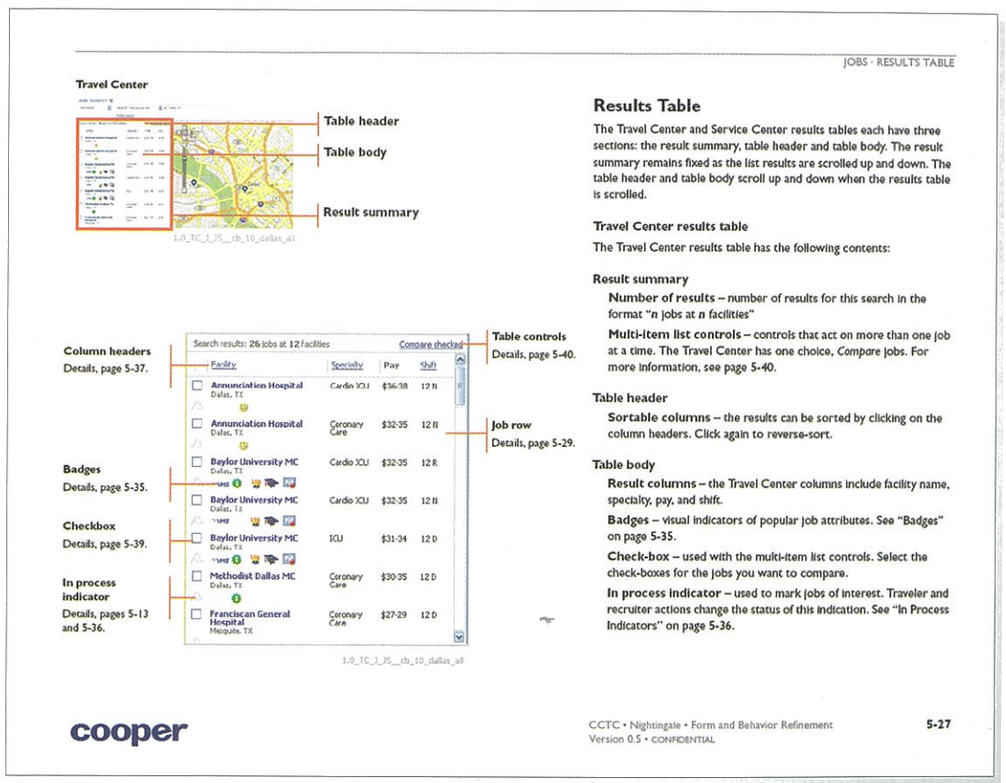
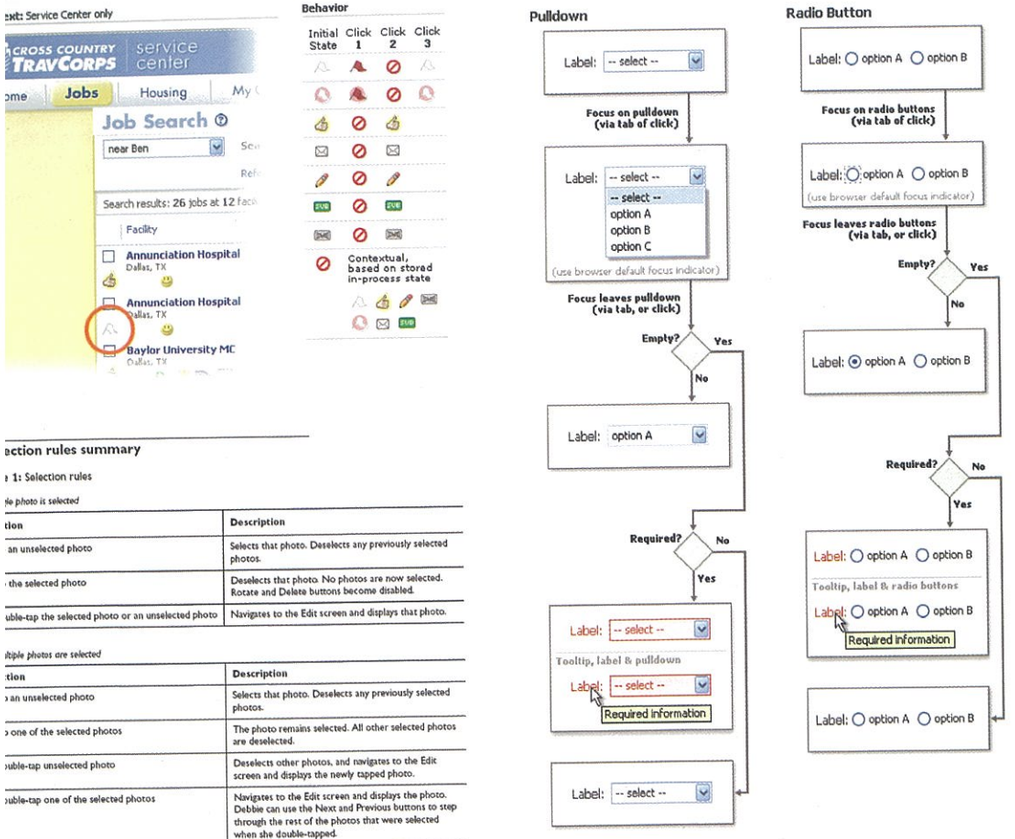


Figure 24.7. An overview of the results list for Cross Country's job search function. Note how the callouts refer to subsequent pages for details



gure 24.8. Examples of useful spec details.

ommon widgets—such as how autocomplete orks in a combo box—if you're using a standard-ed widget library. However, you should certainly escribe any novel widgets or deviations from tandard widget behavior. You could document ne details about each widget in each place it ppears, but this is inefficient and may lead to nconsistencies in your specification. Instead, use cross-reference or link to point to the widget ection of the style guide (discussed later in this hapter). This will let you limit your screen-specific ocumentation only to the widget's values and

EXACT MEASUREMENTS AND COLORS

You need to provide exact pixel measurements and color specifications for each element (see Figure 24.15). Ideally, you should also list the file names and locations of any production-ready assets such as icons or photos. You can provide this information either in the context of the individual screen or in a separate visual system section with links or cross-references from the behavior description. Putting these specs in the individual screen section lets the programmer

information without flipping back and forth. However, on most engineering teams, the people building the screens are not coding the behavior at the same time, or even at all. Integrating these details into the body of the F&BS also requires more coordination between the visual designer and IxD synthesizer, and time is often in short supply at the end of a project. For both of these reasons, leaving detailed visual specs in the visual style guide section is usually more successful, but ask your engineering team how they work before you make the decision.

Visual system or style guide

The contents of the visual style guide section include:

- What a visual system is
- Use of the corporate identity for hardware and software
- On-screen color palette and hardware color, materials, and finishes
- Type specifications for hardware and software
- Photography guidelines
- Icons for hardware and software
- Widgets
- Layout
- Individual screen specifications, if not provided in context earlier

The term "style guide" is somewhat overburdened because many companies use style guides instead of form and behavior specifications (and therefore instead of interaction design). The theory seems to be that as long as the engineers all line up their field labels and draw their buttons the same way, usable systems will follow. If you've gotten this far in this book, I don't need to tell you this is a false hope.

However, a visual style guide is still a useful tool for exposing shortcomings in the visual system, just as the rest of the F&BS helps ensure the completeness and coherence of interaction design. It also helps ensure consistency in screen rendering within a single design team, across multiple design teams working on related products, or when a new designer continues work started by another.

Although the sequence of the style guide subsections is not critical, it's a good idea to put the corporate identity information first, both to acknowledge the importance of the brand and to explain why some aspects of the visual system may deviate from print identity guidelines. Individual screen layout specs, if included in this section, generally come last simply because there are so many of them. The color palette and type are incorporated into widgets and text, so these are generally placed right after the identity discussion.

USE OF THE CORPORATE IDENTITY

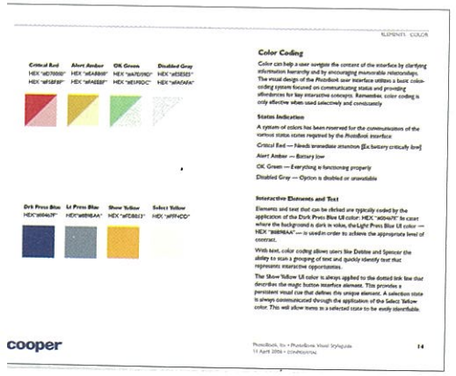
Print and even Web guidelines for the use of the corporate identity sometimes have to be modified for use in applications, in which a mark might be reduced to a tiny icon in a window title bar. The specification needs to show how the identity can degrade gracefully at lower resolutions without becoming debased entirely. For example, if the image and word mark that make up the identity are never separated in any other materials, the corporate brand stewards would have good reason to feel uncomfortable with you using the image alone as an icon. However, since most companies don't have brand manuals that consider software or hardware, a translation (in close cooperation with the corporate branding team) is nearly always required.

Document the fundamental aspects of using the mark in an application or applied to hardware. These include:

- Ensuring that it is represented using the closest digital approximation of your identity colors
- Maintaining its aspect ratio and the spacing between image and word mark
- Ensuring a certain amount of breathing room so the mark is accorded a special status, not crowded in amongst a bunch of controls

OLOR PALETTE

ne color palette for an application or Web site enerally begins with the primary and accent and colors (or the closest digital approximation), upplementary colors for interface elements, and ny colors reserved for color coding and visual edback (such as red, yellow, and green). If a imary identity color is not particularly visible the interface, explain why heavier use of that lor wouldn't have been a good idea from a usility point of view. Explain the intent and any age guidelines for color coding, such as, "Use d only for problems so severe they will prevent e analysis from running," or, "Use 'selection ue' only for selected items and default actions." ecify colors using RGB or hexadecimal numrs, depending on the development platform and gineer preferences, as shown in Figure 24.9.



ure 24.9. A typical page specifying part of an applica-'s color palette.

CMF (color, material, and finish) specifications for hardware are generally delivered to manufacturing separately from the rest of the F&BS—for which manufacturing has no use—but are useful to repeat in the spec just so everything is in one place. Most often, these specs consist of hardware renderings with callouts specifying material, color, and intended finish (see Figure 24.10) using a system such as Pantone's plastic color system or an automotive paint system for metallics. If the designer and manufacturer don't use the same proprietary color system, the color "spec" may consist of a set of physical samples the manufacturer needs to match. A model making vendor usually develops these under the industrial designer's supervision.

TYPE SPECIFICATIONS

Specify which fonts, sizes, and styles should be used for various purposes. As with color, explain why any deviations from the brand manual are necessary, such as why the primary corporate face(s) would be a poor choice for most content and why your choices are appropriate. Explain the type sizes and treatments for each level in the hierarchy, when it is appropriate to use each level, and which sizes should or should not be anti-aliased.

Figure 24.10. Color and material specifications for hardware.

Specify how type is aligned (generally along the left side and on the baseline). Explain any conventions you've established regarding capitalization (such as sentence caps or title caps) and punctuation. For content, specify line spacing (leading), paragraph spacing, and any special spacing for headers. Figure 24.11 shows excerpts from typical type specifications.

Explain why any deviations from the brand manual are necessary.

Include text for hardware labels and such in this section, as well. The manufacturer will need this information delivered separately, but it's useful to see how text used on the surface of a device relates to the rest of the system.

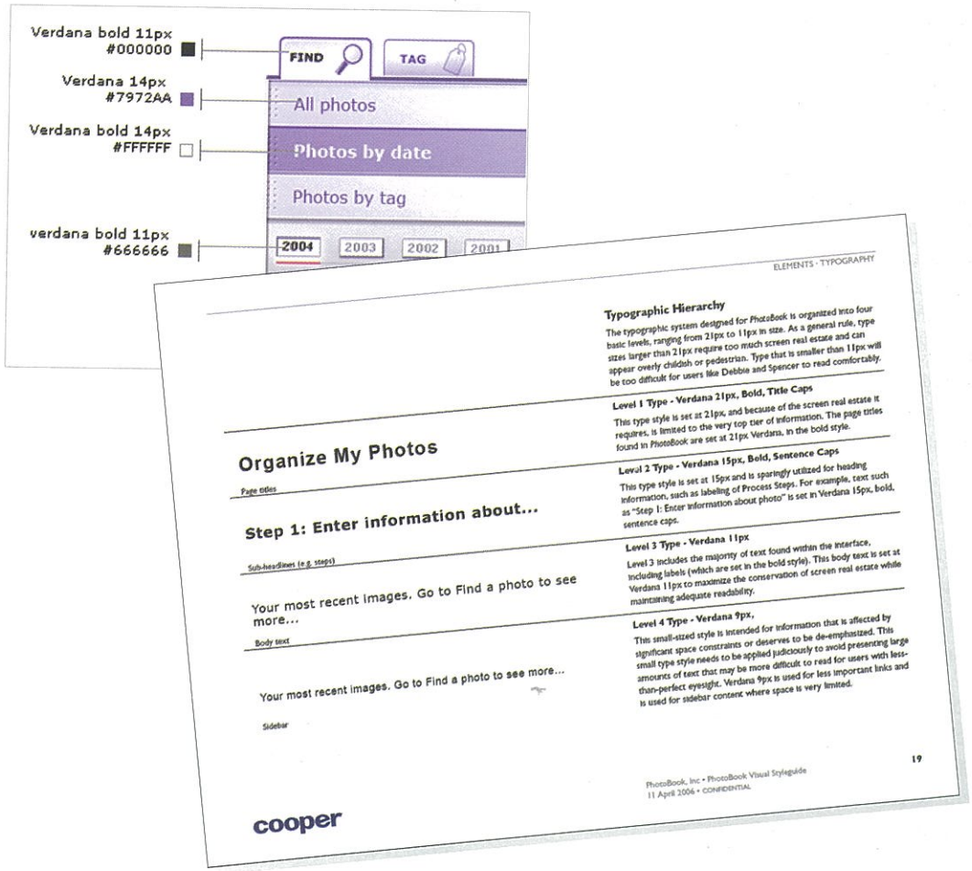


Figure 24.11. Typical type specifications.

PHOTOGRAPHY GUIDELINES

Where photographs are part of a Web site or other product, they are critical to establishing a cohesive style; for example, soft-focus color images photographed at oblique angles don't usually mix well with sharp, head-on, black-and-white images. Unlike icons and other interface components, Web site photos may be added to or updated frequently. Use the experience attributes to explain what the images are supposed to express. Refer to a corporate style guide as appropriate. Be specific about any photographic style issues such as angle, lighting, and depth of field. When appropriate, also specify content, such as:

- Whether images always have people or other specific content in them
- Who the models are (e.g., what range of diversity they should represent, whether they should look average or beautiful, etc.)
- What models are wearing (e.g., solid colors, conservative business clothes, or rumpled casual clothes)
- What the models are doing (e.g., looking at the camera or involved in some natural-looking behavior)
- What emotions the models should convey (e.g., pleased but not grinning)
- What the background is like (e.g., neat, cluttered, indoors, in nature)
- Any signature color palette (e.g., earth tones, bright colors, etc.)

Beyond this, specify any instructions for image processing to achieve a specific, consistent look, such as applying a specific Photoshop filter. It's useful to provide a template file, as well.

ICONS

Describe any general principles and conventions you've adopted, such as using one color scheme for most applications and another for utilities, always using object-plus-action, or always using two-dimensional schematics rather than photorealism. Explain the rationale for the icon style both in terms of usability and the experience attributes.

Categorize the icons by function, such as object icons versus action button icons, as shown in Figure 24.12. Show each icon in every necessary state and size. Include any hardware icons as a category, as well. These are delivered separately to manufacturing, but are useful to relate to the rest of the visual system.

Assuming you're providing digital assets, note the file name of each and, in any online documentation, provide a link to its location.

WIDGETS

Any common controls—widgets that aren't unique to a single screen—should be explained here. Engineers don't rebuild widgets for every screen, so it's unnecessary to put these specs in the individual screen documentation. Show all possible widget states (such as selected, unselected, unavailable, rollover, or default). Specify any detailed behavior such as the maximum number of lines in a drop-down before it scrolls, whether and how a combo box autocompletes text, and so forth. Describe screen-specific behavior in that context rather than here. Figure 24.13 shows an example.

LAYOUT

The layout section explains the purpose of the grid and provides an overview of the horizontal and vertical divisions at the macro level. Often, you can export interface images with your guides superimposed, so much of the drawing work is already done. This section also describes what happens when a window is resized or someone has a lower- or higher-than-expected resolution, if applicable. See Figure 24.14.

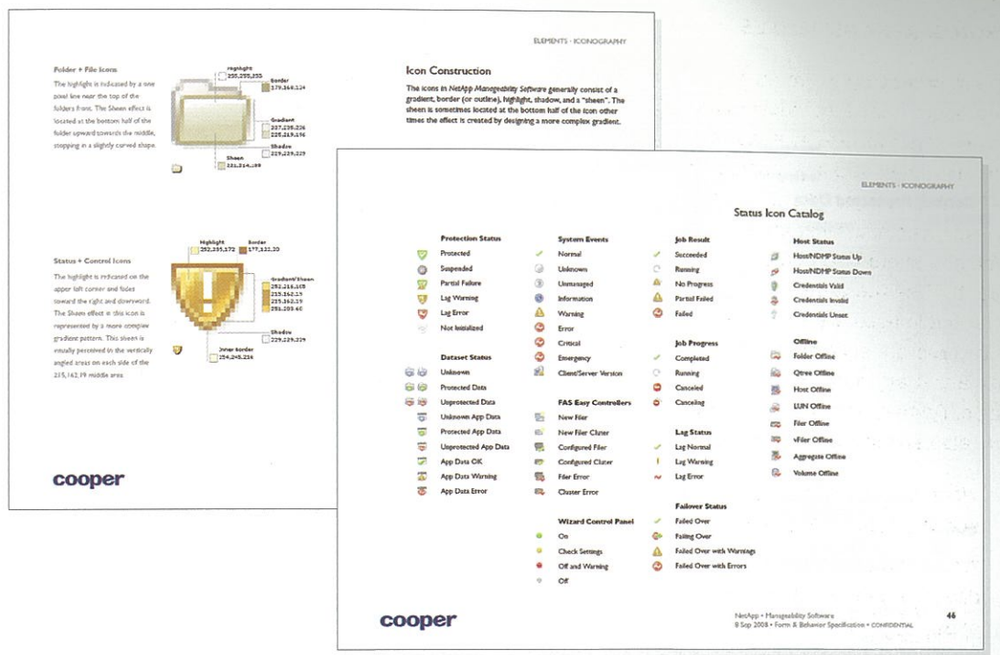


Figure 24.12. Pages from a typical icon catalog.

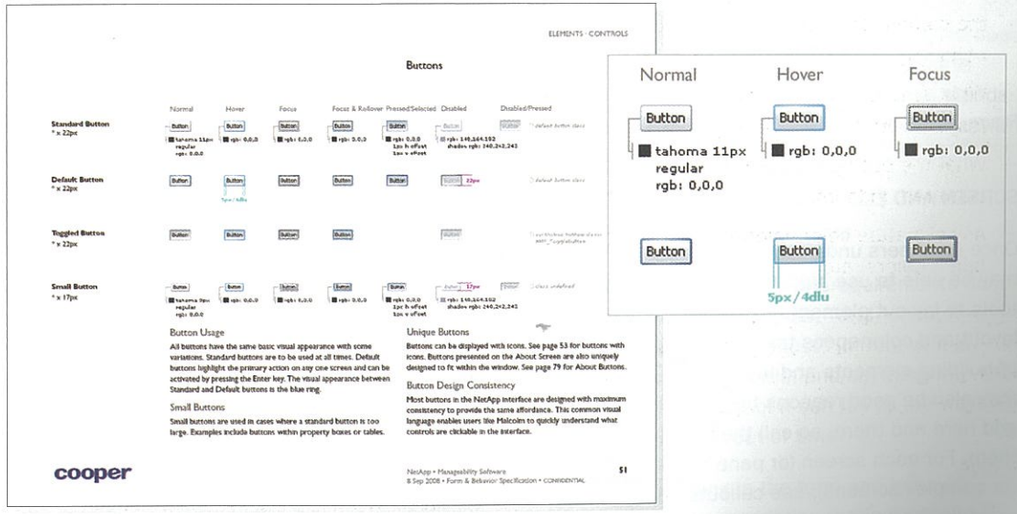
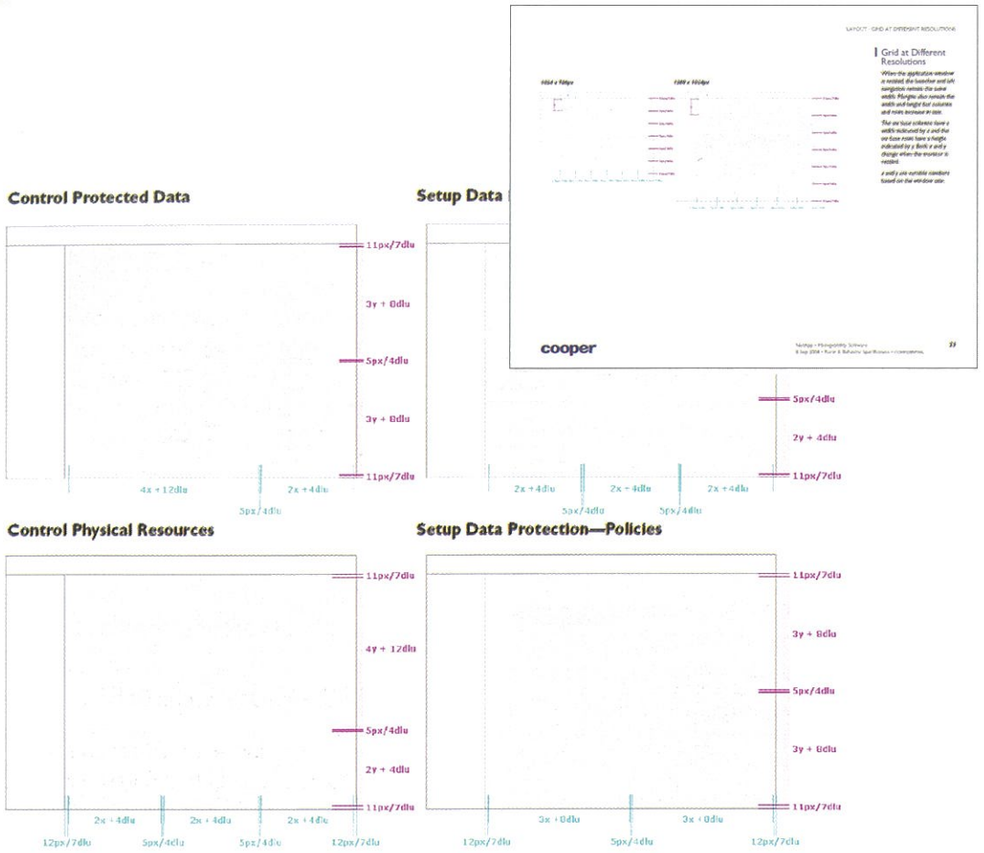


Figure 24.13. Part of a typical widget specification.



igure 24.14. An explanation of the application's macro layout grid and resizing behavior.

CREEN AND ELEMENT SPECIFICATIONS

nce engineers understand the macro grid, they ay be able to use guides in their development ols to lay out elements. However, detailed ayout and color specs take the guesswork out f building elements and laying them out. There ay also be good reasons to deviate from the id here and there, so call these out and explain em. For each screen (or pane or smaller area r complex screens), use callouts and measure- ent lines to specify spacing, color, and file ames for any incorporated images, such as icons or gradients. Figure 24.15 shows a typical spacing and color spec.

Ways to expand or cut back: the F&BS as a product roadmap

Most often, the form and behavior specification documents the next release of the product: whatever is getting built in the near term. Every now and then, though, the F&BS is used for release planning: what fits in the next version and what comes after that. If you're a consultant, the

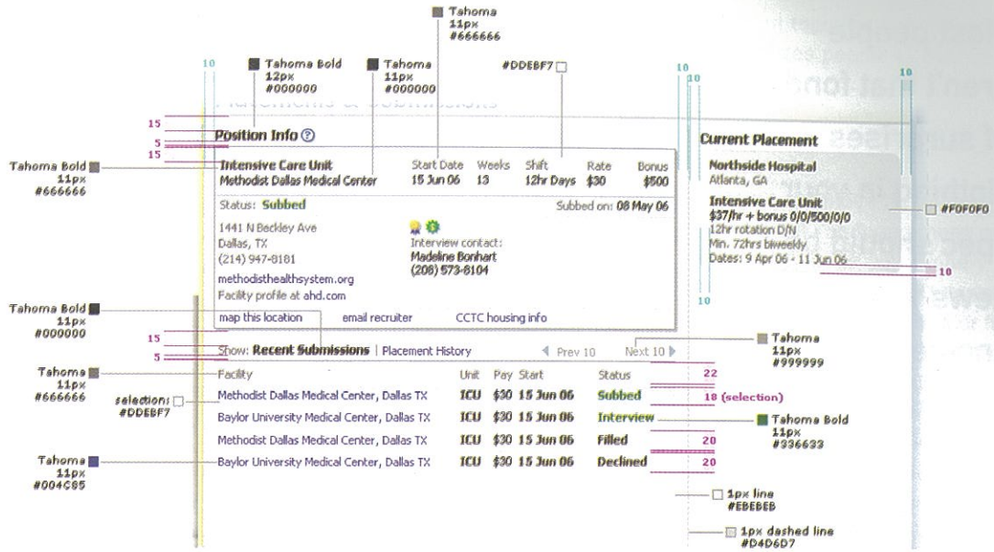


Figure 24.15. Example spacing and color specifications.

happen without you, so you may want to throw in a few longer-term ideas or ways to step back from the current design in case it proves harder to build than expected.

If you expect the product will grow without you, it may be useful to include a section on how to extend the design, such as what kinds of additional data and functions should (and should not) be added to various parts of the interface. You can also include specific ideas for fancier functionality throughout the text, designating these ideas with a light bulb icon or other special treatment to indicate that they're not a near-term specification.

If you think the engineers may have a harder time building the application than they expect to, you can also add some thoughts—either in a single section or woven throughout the document—about how to scale back. Indicate the aspects of the design that you believe are most critical to meeting the personas' needs and the components that could be temporarily replaced by a

Qualities of an Effective Spec

The effectiveness of your specification is not determined only by its content and structure, but also by how you express the content. A good spec is prescriptive, clear, explanatory, efficient to read, and appropriately formatted. It also stands alone, with no need for readers to be familiar with earlier documents.

Prescriptive, not suggestive

Many product managers write requirements documents using the terms "must" and "should"—anything with the word "must" in front of it is presumably nonnegotiable, and anything with the word "should" in front of it can fall by the wayside if the engineers run out of time.

However, everything in the form and behavior specification is a prescription that's meant to be implemented, not a collection of suggestions. The

Most people aren't that fond of surprises. Nothing in your spec should be news to the engineering team.

language should not imply that they are. Write with confidence and conviction about the product and the people who will use it. Avoid saying what the product should, could, or might do; say what it *will* do.

Clear and professional, not pretentious

The form and behavior spec is a professional business document, so appropriateness, organization, grammar, and spelling all count. If you're sloppy about these, some readers will wonder if you were similarly sloppy in your design thinking. As discussed in Chapter 13, use active voice and simple sentences, and don't feel compelled to show off your prodigious vocabulary. Have a good copy editor take a look at it. Likewise, ask your teammates to give the document an editing pass; see the "Technical review and document QA" section later in this chapter for guidance on this topic.

Unsurprising

Aside from, say, getting a fabulous gift or a bigger tax refund than expected, most people aren't that fond of surprises. Engineers who have to build impossible things on tight deadlines are less fond of surprises than most. Nothing in your spec should be news to the engineering team—they should have seen it all before, at least in the form of a whiteboard sketch. Of course, some things are more alarming in a final spec than in a whiteboard meeting because they're more detailed. This sort of thing may be impossible to avoid entirely, but constant communication and incremental delivery of specification chunks can keep it to a minimum.

Persona-focused

People who understand why difficult or unpleasant things are important are more likely to do those things. Engineers and product managers are no exception. Even though there should be no surprises in the spec, the engineer coding it three months down the road (and two days before his deadline) may not recall the rationale for that custom widget. It's best to make sure that rationale is clearly documented, especially if you're not always sitting right across the room to provide it on a moment's notice.

Throughout the document, refer to your personas by name to keep them present in everyone's mind. In the product, interface, and screen overviews—and as needed for custom widgets or other tricky

bits of the design—describe how the solution benefits the persona(s). Sidebars are a useful way to do this within the detailed parts of the specification.

Standardized

A template used by your whole organization helps streamline document creation. If you know what sections belong where and you have page templates for typical layouts, you can focus on communicating the content instead of on how large your margins should be or how a level two header should look.

A standard template also gets engineers accustomed to looking for certain things in certain places. Although some engineers may have idiosyncratic preferences in documentation, it's not productive to vary your documentation for every engineer. However, some aspects of document structure and style, such as whether page layout specs are separate from behavioral specs, absolutely should flex depending on how a particular engineering team works.

Arriving at a spec template that makes sense for most people on most projects will take some collaboration among members of the design group and the engineering group. Once you've arrived at a workable template, try it out on a few projects and then check in with everyone about how it's working. Revisit the template every couple of years to see if it's falling behind the times.

Effectively formatted

Most specifications are, to put it kindly, not something people want to pick up and read. This is understandable; a giant wall of text occasionally broken up by headers or wireframes is certainly unappealing. An effective spec design, like an effective product design, finds the right balance between what users (mostly engineers) need and what can be put together in a reasonable time. It also encourages people to pick up the document and peruse it.

SIZE AND ORIENTATION

Many design firms have a fondness for printing large-format documents, but I've found these are sort of like the proverbial greased pig: slippery and hard to handle. They tend to slide off desks and are too big to fit in typical bookshelves or file drawers. The engineers will thank you if you stick to standard A4 or 8½ x11" paper, as will anyone who tries to print the file later without access to a large-format printer. Spiral binding is ideal if the document is small enough, since pages can be folded over completely and the document will still lie flat on a desk. Three-ring binders are usually the best option for large documents and for large teams, where individual engineers only care about certain sections.

The most common objection to standard size paper is that you can't print large images on it. However, if the orientation of the page matches the orientation of most of your images, you can make good use of the real estate you have. This means that if you're doing mobile phone applications, a portrait orientation is fine, but if you're doing desktop apps, a landscape document will let you use much larger screen shots.

LAYOUT AND WAYFINDING

Like a good screen layout, an effective document page layout relies on a grid that guides the placement of text columns, images, headers, and footers. Four page formats cover most of what you'll need in a desktop application spec: a full-page image with room for callouts, a single column of text with one or more images, two columns of text, and a set of two or three screens side by side for a scenario. All four are shown in Figure 24.16.

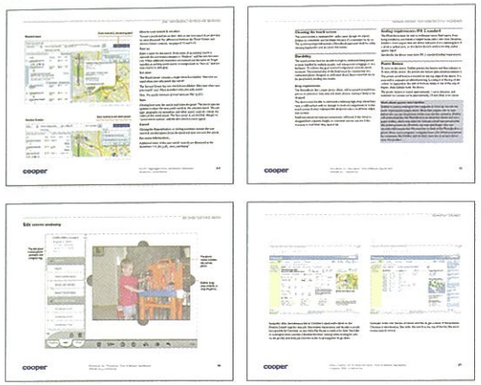


Figure 24.16. Three or four page layouts cover most application spec needs.

Also keep readers oriented by:

- Using consistent section headers at the top of each page
- Using thumbnails of the whole screen if you're zoomed in on a single component
- Repeating the same image on each relevant page if text spans multiple pages
- Making liberal use of cross-references

TEXT FORMATTING

Remember to format content for flippers as well as readers (see Chapter 13); tabs or colored section breaks can help with this in a large document. This is especially important considering that most readers will use the spec as a reference document rather than reading through it from start to finish. A quick skim should acquaint readers with the purpose and scale of the product. A casual read—by a busy executive, for example—should provide a sense of the product's scope, complexity, and benefits. A detailed review should answer nearly every question necessary to produce a technical specification for the product; I say "nearly" because there's often some remain-

You can make the document easier to scan if you format content according to its purpose:

- For sequences, use numbered lists
- For lists of related items or ideas, use bullets
- For comparisons or tabular information, use tables
- For narrative, use paragraphs of text

If you can convey ideas without making people read complete sentences, do so.

Documentation Process and Practices

Creating a detailed form and behavior specification is time-consuming; it requires about one day of writing and drawing time per three days of detailed design meetings, plus a couple of days at the end for editing and proofreading. However, this doesn't mean documentation happens only at the end—it needs to be an ongoing process for multiple reasons. Documenting design as you go helps:

- Ensure its completeness and effectiveness
- Ensure that everyone on the team has a shared understanding as the design evolves
- Avoid a big time crunch at the end of the project
- Avoid forgotten or incorrect details
- Deliver incremental specs to engineers and SMEs for detailed review

The interaction designers generally spend five or six hours a day in detailed design meetings, with two or three hours of solo work time. While the IxDG renders screens and works through issues with the visual designer, the IxDS translates meeting notes and sketches into draft documentation. Even if the design evolves later—and it often does—it's easier to adjust the documentation than to write the whole document at the end of

Documenting as you go

Documentation should be an ongoing process, not something that happens at the end.

Start by outlining your document; either do this as a team or review your outline with your team before investing too much writing time. You should be able to do a fairly thorough outline on the first or second day of the phase based on the topics in your work list and calendar.

Page templates make it easy to start dropping in content, even if it's rough. Copy any reusable sections from your user and domain analysis document. If you find yourself staring at blank pages, start laying out at least the overview pages by dropping in your framework sketches and notes.

As you begin to have design meetings, add your meeting notes to the document each day. It's better to leave things as bullet points for now than to write sloppy paragraphs, which have a way of sneaking into the final document. Add empty headers where you don't have information; these may also serve as reminders for topics you still need to cover in the next design meeting.

It's useful to paste in digital photos of whiteboard sketches as placeholders for screen drawings. This lets you start thinking about page layout and makes it easier to visualize and explain the design. Some people find it easiest to do the anatomy callouts for the sketch, then use those to generate a set of content headers for subsequent descriptions. However, don't add lines pointing from the callout text to the image yet; you'll save yourself a lot of tweaking if you add these after the visual design is finalized.

It's likely you will find unresolved issues in the design as you write. If the answer seems straightforward, go ahead and write down the behavior that seems most appropriate, but highlight this text in another color so you'll remember to discuss this with your design partners. Format open issues in some color that stands out, as well. Circulate drafts to your teammates to make sure everyone has a shared understanding of the design.

Whether you're delivering the design topic by topic (which is ideal) or all at once, you'll need a bit of time to turn your bullet lists and sentence fragments into a coherent document. For most people, the first pass is about making sure the content is complete and readable. A second pass allows for polishing, adding lines between callout text and components, and inserting any missing cross-references.

Managing images

Drawing screens can be even more time-consuming than writing about them, particularly if you try to illustrate every screen in every state. Instead, identify what drawings will give you the most leverage; for example, if you need to show what a list box looks like in the open state, can you reuse an image you drew for a scenario rather than drawing a separate image for that purpose? A detailed picture list makes it easier to minimize the number of unique images you'll need.

As you finish each design topic at the whiteboard, identify what screens and components you need to depict for the anatomy and behavior to be clear. After that, develop a detailed version of your key path scenario, including selection states and any specific data to illustrate; this will help the IxDG put the right contents in the screens. Determine how few images you can use to tell the story; what can you describe clearly enough with just text? Finally, see how many of the images from your story can be recycled in the anatomy section.

As the IxDG and visual designer update screens, they should be dropping exported images into a consistent folder. As long as the new images are saved with the same names and in the same location, you can import them by reference, which saves a lot of cutting and pasting time and minimizes versioning errors.

Technical review and document QA

It's essential to make sure teammates agree with what the documentation says, but you'll also benefit from having someone edit your documentation to ensure that:

- The design is correctly described.
- Engineers will have all the information they need.
- The organization makes sense.
- The document is clearly written.
- You haven't unintentionally stepped on anyone's toes.
- You've explained the reasoning behind design decisions, especially any that deviate from conventions or are difficult to build.
- You've used the personas throughout the document.
- Every state illustrated in the screens is correct.
- Components are correctly and consistently labeled.
- Callouts point to the right things.
- Cross-references are correct.
- Data in screen shots is free from potentially distracting errors in spelling, calculation, etc.

However, reviewers shouldn't try to rewrite the document in their own voices; as long as the way you've written it is clear and unlikely to offend stakeholders, that's what's important.

Circulate drafts early. If you can, have your teammates sequentially mark up the same copy of the document; this keeps them from spending time on the same edits, surfaces any disagreements, and gives you a chance to make updates from one place rather than trying to work through two or three separate sets of comments. Let reviewers know if you're not ready for detailed proofing and only want comments on content. As you get comments back, use a highlighter to mark off edits as you go (or note that you're ignoring them) so you're sure you've got them all. If you disagree with a significant comment or edit, discuss it with your teammates rather than disregarding it.

Documentation tools

Microsoft Word is unsuitable for large specs because it doesn't allow terribly precise control of page layout, and it may not be quick or stable with very large documents. Cooper started using Adobe FrameMaker, which is geared toward large, complex technical documents, around 1999. It allows for cross-referencing, better control of images, and a number of other useful behaviors. However, the learning curve is steep, and there are some ways in which the application is inflexible because it's aimed more at documenting finished systems than at documenting a design as it evolves. It's always been a love-hate relationship at best.

Now that there's finally a stable cross-referencing plug-in for Adobe InDesign, we've begun using that for most documents. No doubt we'll find it involves some trade-offs of its own, but it seems to be working well so far.

Regardless of what tool you choose, look for something that:

- Allows you to import images by reference instead of pasting
- Provides automated cross-referencing
- Lets teammates and reviewers mark up soft copies without mastering an obscure tool
- Handles multiple page-layout styles in the same document
- Allows precise control of image and text position
- Is stable and not too slow when the file gets large

Presenting Detailed Design

Formally presenting the detailed design provides stakeholders with an overview of the design, gives you an opportunity to show off what you've accomplished, and gives everyone yet another chance to ask questions. This can happen in a single, large presentation near the end of the project or, if possible, in several smaller presentations as you work your way through various design chunks.

Detailed hardware design often involves more than one formal milestone. A discussion of the appearance model(s) may still result in some evolution, especially if you've done two or more models to help finalize the design language or some other aspect of the form. The final delivery of a prototype (which could take place during the support phase rather than detailed design) lets stakeholders assess every detail, though this is primarily targeted at the engineering and manufacturing audience.

For nearly any product, it's useful to split your presentation into two sessions: an hour or two for a general audience of all stakeholders, followed by a detailed review with the engineers, any SMEs, business analysts, or manufacturing reps, and perhaps the QA lead. This lets you cover the high points with the general audience while giving the people who need it a chance to examine the minutiae.

Structuring and delivering a stakeholder presentation

An effective presentation for a mixed audience is a lot like the framework presentation (see Chapter 19), except with bitmap screens, no separation of visual and interaction design, and fewer supporting arguments for the overall structure (since most decisions have already been made). If you present the scenarios plus what's in the product, framework, screen-level overviews, and a hardware overview with an appearance model (if applicable), you'll be at about the right level. A typical outline looks like this:

- Project background
- Quick persona recap (more detailed if new stakeholders are present)

- Product and interface overview, including hardware and design language
- Scenarios
- Benefits (how the design serves the personas)
- Hardware detail and interface anatomy slides (for those who care to stay and discuss)

The product and interface overview helps remind stakeholders of major decisions made to date. A quick overview of the interface anatomy and a review of the hardware design set the stage for the scenarios.

Although the appearance model serves as the main focal point for discussion of the hardware function and physical design language, photos of the appearance model are useful references, especially for any remote stakeholders who cannot see the model. The ID typically passes the appearance model around the room toward the end of the hardware discussion (with an appropriate caution about its fragility).

Notice that details such as screen-level anatomy and hardware assembly details are at the end. This allows stakeholders to step out of the meeting if they don't need to understand the details. It's usually effective just to show a full-screen slide of each major view as a visual aid for discussion of whatever the audience cares about. You might also want to include zoomed-in views of any complex elements that are especially important.

In most ways, delivering a detailed design presentation differs very little from delivering more conceptual work. You might get a few more comments that someone doesn't like this or that icon, but overall, the guidance in Chapter 19 should serve you well in explaining and defending your decisions.

Comprehensive walkthroughs

Hardware and software engineering teams are certain to have different questions and interests so you may want to split the comprehensive discussions into separate groups for convergent products. The detailed software discussion requires the interaction designers and visual designer. The ID may be the only team member required at the hardware discussion.

SOFTWARE REVIEW

For the detailed software review—which can take anywhere from a couple of hours to a couple of days, depending on how big a chunk of design you're delivering—a slide deck is often not the best tool. It's typically more useful to hand out the documentation (if it exists in hard copy) and to project the document pages up on the screen as you discuss them. Engineers may fail to read even the best-designed documentation, so a thorough introduction will help everyone see that the answers to nearly all of their questions are easy to find.

This meeting is usually more of a facilitated question-and-answer session than a presentation, though it's a good idea to start out with an overview of the document and its contents. The questions and discussion are likely to require the expertise of every design team member. However, the meeting could get chaotic, so one person should still be the team's primary facilitator, asking participants to hold a question for later, writing down open issues, and referring questions to teammates as needed. An experienced IxDS is often best suited to this role, both due to a natural inclination toward facilitation and to a deep familiarity with the documentation.

HARDWARE REVIEW

During an appearance model review, the ID's goals are to make sure the engineers and manufacturer understand the design intent and to minimize engineering decisions that will unnecessarily compromise that intent. This is the first op-

detailed physical representation of the design and determine whether adjustments are needed.

The ID and ME tend to share the stage during this discussion. The ID focuses on the aesthetic and usability issues, including color choice, material selection, finish strategy, visible part breaks, and the behavior of physical controls or mechanisms. The ME may project a CAD file on the screen to discuss internal components, part count, and overall manufacturing strategy.

A prototype review (which could happen during the support phase, also) is similar. The mechanical, electrical, and manufacturing-focused engineers can disassemble the prototype to examine every tiny detail of how components fit and attach. This discussion is focused primarily on tooling and manufacturing issues; the design intent is presumably understood by this point, so this meeting is primarily the ME's show.

Summary

Communication about detailed design, although it may result in a single, massive specification, should never be a single event. Informal communication needs to happen with engineers, business analysts, and subject matter experts on a regular basis. Multiple instances of formal communication are ideal if you can document your design in chunks, since this gives the appropriate collaborators a chance to review the finer nuances of your work.

Regardless of the frequency of formal communication, be sure to include plenty of detailed images and explanatory text. Be prescriptive: What goes in the spec needs to be exactly what gets built, except to the extent that the design and engineering teams need to modify the spec together for cost or timing reasons. However, never surprise the engineering team with anything new in the spec; it might make your delivery a bit anticlimactic, but a product that gets built is far more rewarding than having an audience surprised by your brilliance at a final presentation.