O'REILLY®

4th Edition

Information Architecture

FOR THE WEB AND BEYOND

Louis Rosenfeld,
Peter Morville & Jorge Arango

Information Architecture: For the Web and Beyond
by Louis Rosenfeld, Peter Morville, and Jorge Arango

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Mary Treseler
Editor: Angela Rufino
Production Editor: Matthew Hacker
Copyeditor: Jasmine Kwityn
Proofreader: Rachel Head

Indexer: Judith McConville
Interior Designer: David Futato
Cover Designer: Ellie Volckhausen
Illustrator: Rebecca Demarest

February 1998: First Edition
August 2002: Second Edition
December 2006: Third Edition
September 2015: Fourth Edition

Revision History for the Fourth Edition
2015-09-01: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491911686* for release details.

978-1-491-91168-6

[TI]

Table of Contents

Part II. Basic Principles of Information Architecture

CHAPTER 13

Design and Documentation

You can use an eraser on the drafting table
or a sledge hammer on the construction site.
—Frank Lloyd Wright

In this chapter, we'll cover:

- The role of diagrams in the design phase
- Why, when, and how to develop sitemaps and wireframes, the two most common types of IA diagrams
- How to map and inventory your content
- Content models and controlled vocabularies for connecting and managing granular content
- Ways to enhance your collaboration with other members of the design team
- Style guides for capturing your past decisions and guiding your future ones

When you cross the bridge from research and strategy into design, the landscape shifts dramatically. The emphasis moves from process to deliverables as your clients and colleagues expect you to move from thinking and talking to actually producing a clear, well-defined information architecture.

This can be an uneasy transition. You must relinquish the white lab coat of the researcher, leave behind the ivory tower of the strategist, and delve into the exposed territory of creativity and design. As you

commit your ideas to paper, it can be scary to realize there's no going back. You are now actively shaping what will become the user experience. Your fears and discomforts will be diminished if you've had the time and resources to do the research and develop a strategy; if you're pushed straight into design (as is too often the case), you'll be entering the uncertain realm of intuition and gut instinct.

It's difficult to write about design because the work in this phase is so strongly defined by context and influenced by tacit knowledge. You may be working closely with a graphic designer to create a small website or app from the ground up. Or you may be building a controlled vocabulary and index as part of an enterprise-level redesign of a broad information environment that involves more than a hundred people. The design decisions you make and the deliverables you produce will be informed by the total sum of your experience.

In short, we're talking about the creative process. Ours is a vast, complex, and ever-changing canvas. Often, the best way to teach art is through the time-tested practice of show-and-tell. So, in this chapter, we'll use work products and deliverables to tell the story about what we do during the design phase.

Before we dive in, here's a caveat: although this chapter focuses on deliverables, process is as important during design as it is during research and strategy. This means that the techniques covered previously should be applied to these later phases, albeit with more concrete and detailed artifacts—ranging from vocabularies to wireframes to working prototypes—being tested.

And another caveat: for reasons beyond your control, you'll occasionally—even frequently—find yourself in the uncomfortable situation of bypassing research and strategy altogether, skipping headlong into the abyss of design. Deliverables are especially critical in this context; they're anchors that, by forcing the team to pause, capture, and review its work, regulate and moderate an out-of-control project. You can also use deliverables to unmask design problems and force the project to backtrack to research and design tasks that should have been handled much earlier.

Guidelines for Diagramming an Information Architecture

We are under extreme pressure to clearly represent the product of our work. Whether it's to help sell the value of information architecture to a potential client or to explain a design to a colleague, we rely upon visual representations to communicate what it is we actually do.

And yet information architectures—as we've mentioned many times—are abstract, conceptual things. Websites, in particular, are not finite; often you can't tell where one ends and the other begins. Subsites and the "invisible web" of databases further muddy the picture of what should and shouldn't be included in a specific architecture. Digital information itself can be organized and repurposed in an almost infinite number of ways, meaning that an architecture is typically multidimensional—and therefore exceedingly difficult to represent in a two-dimensional space such as a whiteboard or a sheet of paper.

So we're left with a nasty paradox: we're forced to demonstrate the value and essence of our work in a visual medium, though our work itself isn't especially visual.

There really is no ideal solution. The field of information architecture is too young and dynamic for its practitioners to have figured out how best to visually represent information architectures, much less agree upon a standard set of diagrams that work for all audiences in all situations.[1] And it's unlikely that the messages we wish to communicate will ever lend themselves easily to 8.5″ × 11″ sheets of paper.

Still, there are a couple of good guidelines to follow as you document your architecture:

1 For an in-depth look at deliverables, we recommend Dan Brown's *Communicating Design: Developing Web Site Documentation for Design and Planning, Second Edition* (San Francisco: New Riders, 2010). Dan is an information architect whose work is highly respected by many practitioners.

Provide multiple "views" of an information architecture
: Information environments are too complex to show all at once; a diagram that tries to be all things to all people is destined to fail. Instead, consider using a variety of techniques to display different aspects of the architecture. No single view takes in the whole picture, but the combination of multiple diagrams might come close.

Develop those views for specific audiences and needs
: You might find that a visually stunning diagram is compelling to client prospects, therefore justifying its expense. However, it probably requires too many resources to use in a production environment, where diagrams may change multiple times per day. Whenever possible, determine what others need from your diagrams before creating them. For example, Keith Instone, an information architect formerly at IBM, created very different diagrams for communicating "upstream" with stakeholders and executives than for communicating "downstream" with designers and developers.

Whenever possible, present information architecture diagrams in person, especially when the audience is unfamiliar with them. If you can't be there in person, at least be there via videoconference or telephone. Again and again, we've witnessed (and suffered from) huge disconnects between what the diagram was intended to communicate and what it was actually understood to mean. This shouldn't be surprising, because, as we mentioned, there is no standard visual language to describe information architectures yet. So, be present to translate, explain, and (if necessary) defend your work.

Better yet, work in advance with whomever you're presenting your diagrams to—clients, managers, designers, programmers—so they can understand what they will need from them. You may find that your assumptions of how they would use your diagrams were quite wrong. We've seen a large, respected firm fired from a huge project because it took too many weeks to produce bound, color-printed, sexy diagrams. The client preferred (and requested) simple, even hand-drawn, sketches because it needed them as soon as possible.

As we've seen in previous chapters, the most frequently used diagrams are sitemaps and wireframes, which focus more on the structure of content than its semantic value. Because of this, sitemaps and wireframes are not as effective at conveying the semantic nature of

content or labels. We'll discuss both types of diagrams in detail in the following sections, but first it's helpful to understand the visual language that these diagrams use.

Communicating Visually

Diagrams are useful for communicating the two basic aspects of an information system's structural elements.[2] Diagrams define:

Content components
: What constitutes a unit of content, and how those components should be grouped and sequenced

Connections between content components
: How content components are linked to enable actions such as navigating between them

No matter how complex your diagrams may ultimately become, their main goal will always be to communicate what your information environment's content components are and how they're connected.

A variety of visual vocabularies have emerged to help convey the complexity of information architecture in visual diagrams, each providing a clear set of terms and syntax to visually communicate components and their links. The best known and most influential visual vocabulary is Jesse James Garrett's, which has been translated into eight languages. Jesse's vocabulary anticipates and accommodates many uses, but perhaps the greatest reason for its success is its simplicity; just about anyone can use it to create diagrams, even by hand.

Visual vocabularies are at the heart of the many templates used to develop sitemaps and wireframes. Thanks to their developers' generosity, there are many free templates you can use to create your own deliverables; Table 13-1 provides useful examples. Each requires one of the common charting programs, like Microsoft's Visio (for Windows PCs) or Omni Group's OmniGraffle (for Macs).

2 Semantic aspects, like controlled vocabularies, don't lend themselves as easily to visual representation.

Table 13-1. Templates for common diagramming tools

Name	Creator	Application	URL
OmniGraffle Wireframe Stencils	Michael Angeles	OmniGraffle	*http://bit.ly/omnigraffle_wireframe*
Sitemap Stencil	Nick Finck	Visio	*http://www.nickfinck.com/stencils.html*
Wireframe Stencil	Nick Finck	Visio	*http://www.nickfinck.com/stencils.html*
Block Diagram Shapes Stencil	Matt Leacock, Bryce Glass, and Rich Fulcher	OmniGraffle	*http://www.paperplane.net/omnigraffle/*
Flow Map Shapes Stencil	Matt Leacock, Bryce Glass, and Rich Fulcher	OmniGraffle	*http://www.paperplane.net/omnigraffle/*

What if you're a nonvisual person who cringes at the idea of learning OmniGraffle? Or the people you're communicating your ideas to aren't visually oriented? Does your work *have* to be visual?

Absolutely not. As ugly as the results may be, you can render your sitemaps as outlines in a word processor or use a spreadsheet's cells in a similar fashion. You can write page descriptions that cover the same bases as your wireframes—just about anything can be rendered in text. Ultimately, these deliverables are first and foremost communication tools. You need to play to your own communication strengths and, more importantly, take advantage of whatever style works best for your audience.

But remember, there's a reason they say "a picture is worth a thousand words." The lines between information architecture and the more visual aspects of design are blurry, and at some point, you'll have to connect your IA concepts, however textual, to the work that is the responsibility of graphic designers and interaction designers. Hence, we spend most of our time in this chapter on visual means for communicating information architectures.

Sitemaps

Sitemaps show the relationships between information elements such as pages and other content components, and can be used to portray organization, navigation, and labeling systems. Both the diagram and the navigation system display the "shape" of the information

space in overview, functioning as a condensed map for site developers and users, respectively.

High-Level Architecture Sitemaps

High-level sitemaps are often created as part of a top-down information architecture process (and they may also be produced during a project's strategy phase.) Starting with the main page, you might use the process of developing a sitemap to iteratively flesh out more and more of the architecture, adding subsidiary sections, increasing levels of detail, and working out the navigation from the top down. Sitemaps can also support bottom-up design, such as displaying a content model's content chunks and relationships; we discuss these uses later in the chapter.

The very act of shaping ideas into the more formal structure of a sitemap forces you to be realistic and practical. If brainstorming takes you to the top of the mountain, creating the sitemap can bring you back down to the valley of reality. Ideas that seemed brilliant on the whiteboard may not pan out when you attempt to organize them in a practical manner. It's easy to throw around concepts such as "personalization" and "adaptive information architectures." It's not so easy to define on paper exactly how these concepts will be applied to a specific product.

During the design phase, high-level sitemaps are most useful for exploring primary organization schemes and approaches. High-level sitemaps map out the organization and labeling of major areas, usually beginning with a bird's-eye view from the main page of the website. This exploration may involve several iterations as you further define the information architecture.

High-level sitemaps (like the one in Figure 13-1) are great for stimulating discussions focused on the organization and management of content as well as on the desired access pathways for users. These sitemaps can be drawn by hand, but we prefer to use diagramming software such as Visio or OmniGraffle. These tools not only help you to quickly lay out your architecture sitemaps, but can also help with site implementation and administration. They also lend your work a more professional look, which, sadly, can sometimes be more important than the quality of your actual design.

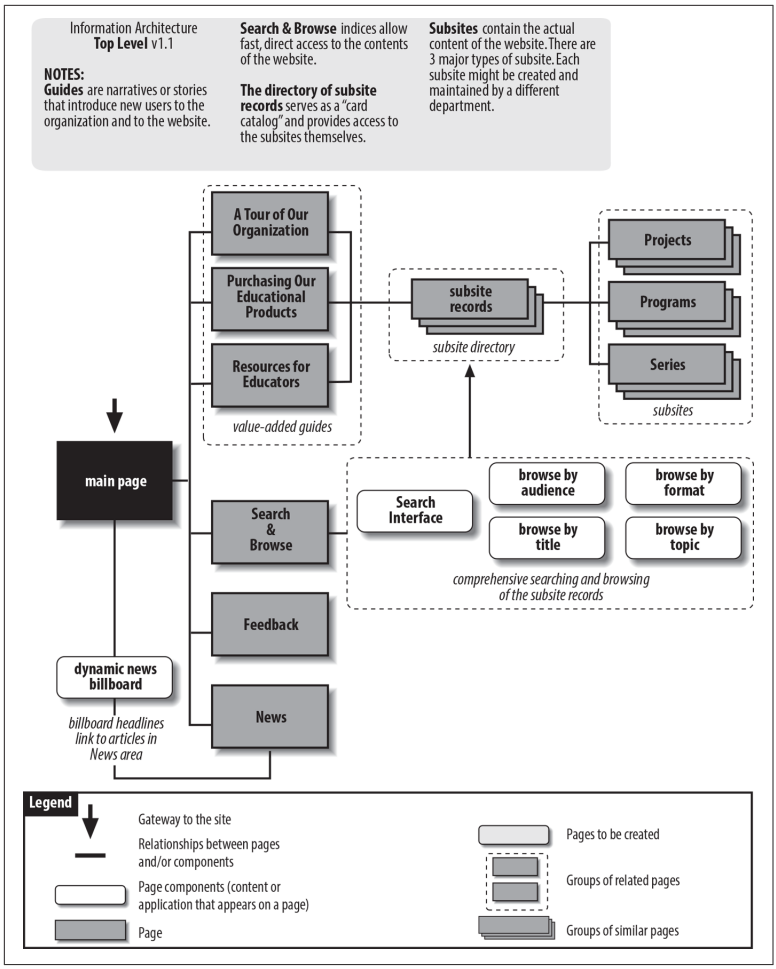

Figure 13-1. A high-level sitemap

Let's walk through the sitemap in Figure 13-1 as if we were presenting it to clients or colleagues. The building block of this architecture is the subsite. Within this company, the ownership and management of content is distributed among many individuals in different departments. There are already dozens of small and large websites, each with its own graphic identity and information architecture. Rather than trying to enforce one standard across this collection of sites, this sitemap suggests an "umbrella architecture" approach that allows for the existence of lots of heterogeneous subsites.

Moving up from the subsites, we see a directory of subsite records. This directory serves as a "card catalog" that provides easy access to the subsites. There is a record for each subsite; each record consists of fields such as *title*, *description*, *keywords*, *audience*, *format*, and *topic*, which describe the contents of that subsite.

By creating a standardized record for each subsite, we are actually creating a database of subsite records. This database approach enables both powerful known-item searching and exploratory browsing. As you can see from the Search & Browse page, users can search and browse by title, audience, format, and topic.

The sitemap also shows three guides. These guides take the form of simple narratives or "stories" that introduce new users to the site's sponsor and selected areas within the website.

Finally, we see a dynamic news billboard that rotates the display of featured news headlines and announcements. In addition to bringing some action to the main page, this billboard provides yet another way to access important content that might otherwise be buried within a subsite.

At this point in the discussion of the high-level sitemap, you are sure to face some questions. As you can see, sitemaps don't completely speak for themselves, and that's exactly what you want. High-level sitemaps are an excellent tool for explaining your architectural approaches and making sure that they're challenged by your client or manager. Questions such as "Do those guides really make sense, considering the company's new plans to target customers by region?" give you an excellent opportunity to gain buy-in from the client and to fireproof your design from similar questions that might arise much later in the process, when it'll be more expensive to make changes.

Presenting sitemaps in person allows you to immediately answer questions and address concerns, as well as to explore new ideas while they're still fresh. You might also consider augmenting your sitemaps with a brief text document to explain your thinking and answer the most likely questions right on the spot. At the very least, consider providing a "Notes" area (as we do in this example) to briefly explain basic concepts.

Digging Deeper into Sitemaps

As you create sitemaps, it's important to avoid getting locked into a particular type of layout. Instead, let form follow function. Notice the difference between Figures 13-2 and 13-3.

Figure 13-2 provides a holistic view of the information architecture for a global consulting firm. It's part of an initiative to build support for the overall vision of unified access to member firms' content and services. In contrast, Figure 13-3 focuses on a single aspect of navigation for The Weather Channel's website, aiming to show how users will be able to move between local and national weather reports and news. Both sitemaps are high level and conceptual in nature, yet each takes on a unique form to suit its purpose.

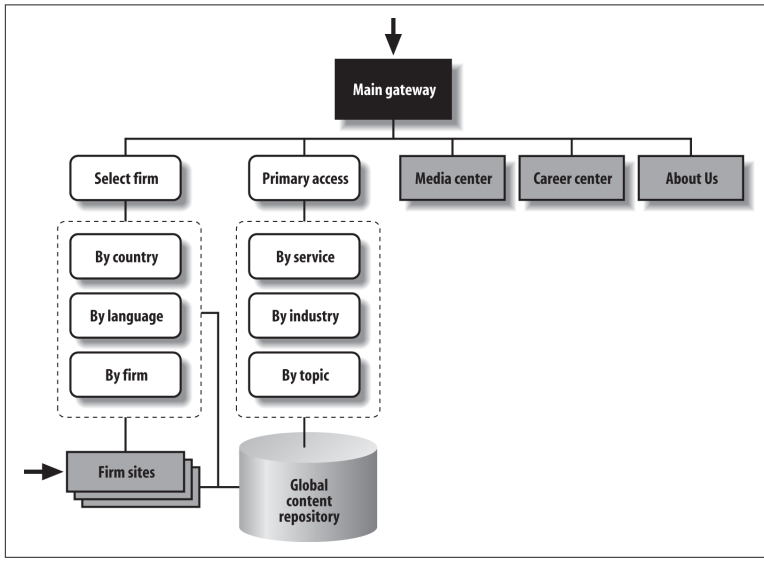

Figure 13-2. This sitemap illustrates the big picture for a consulting firm's public site...

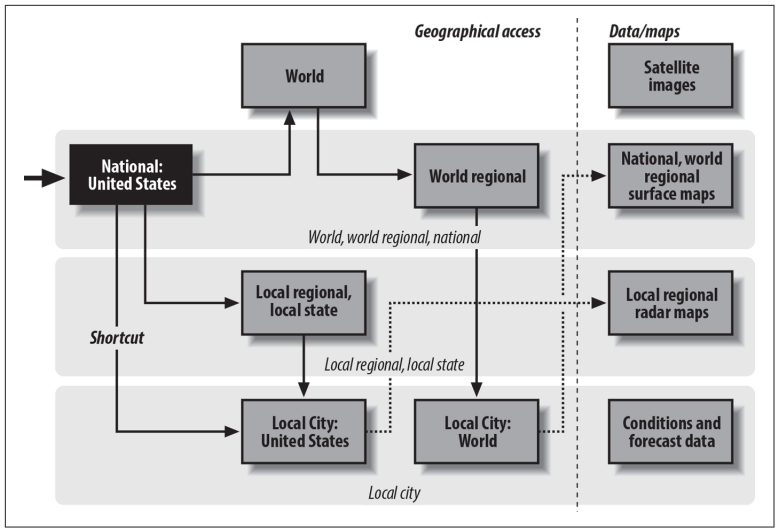

Figure 13-3. ...while this one focuses on geographic hub navigation for The Weather Channel's site...

In Figure 13-4, we see a high-level sitemap for the online greeting card website Egreetings.com. This sitemap focuses on the user's ability to filter cards based on format or tone at any level while navigating the primary taxonomy.

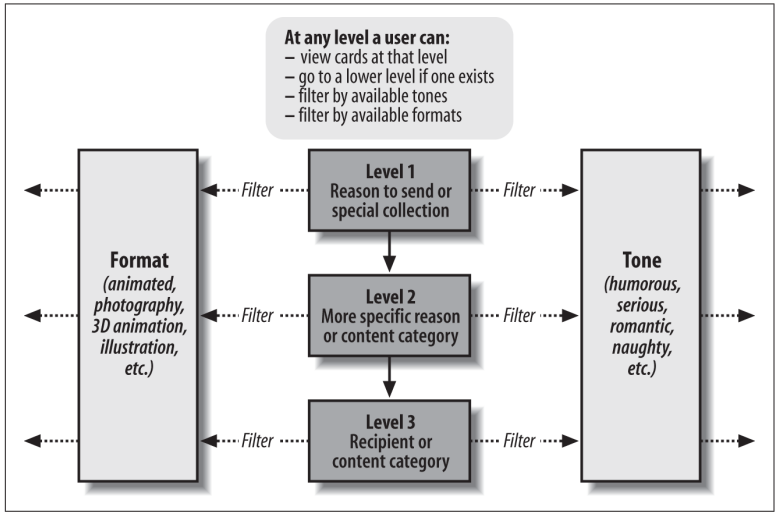

Figure 13-4. ...and this one demonstrates how filtering might work at Egreetings.com

It's important to remind ourselves that information environments aren't just about content; we can also contribute to the design of transactional and task-centered systems. This work requires task-oriented sitemaps and process maps.

For example, Figure 13-5 presents a user-centered view of the card-sending process at Egreetings.com prior to a redesign project. It allows the project team to walk through each step along the web- and email-enabled process, looking for opportunities to improve the user experience.

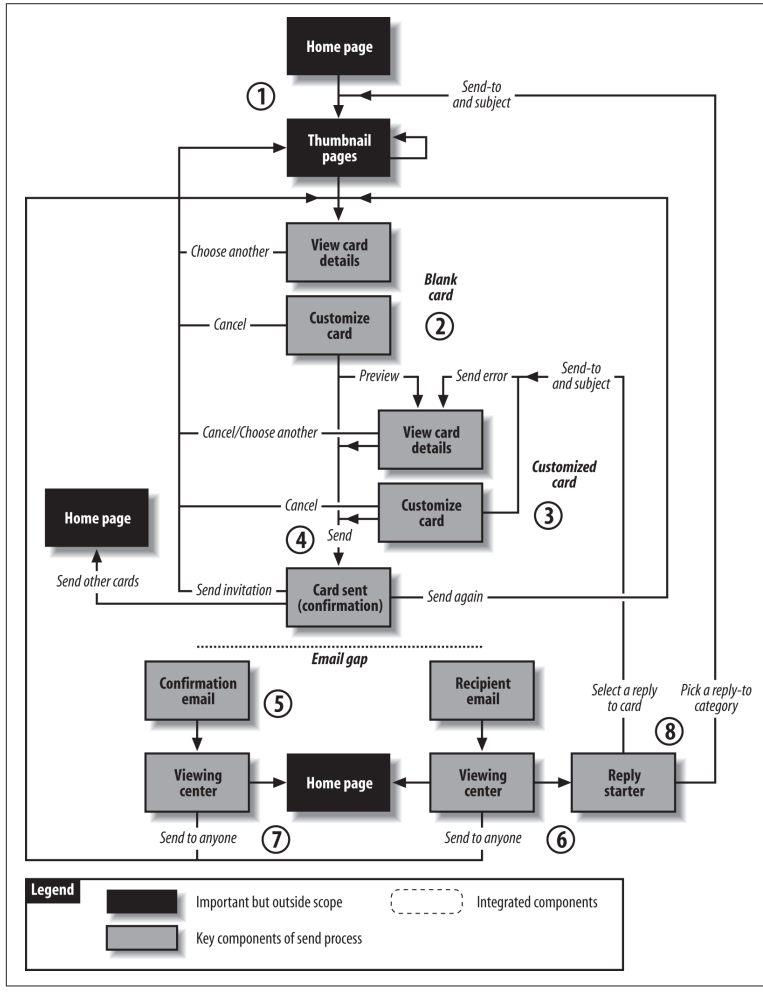

Figure 13-5. A task-oriented sitemap of the card-sending process

Figure 13-6 demonstrates how casual browsers may become engaged in a political campaign over time by interacting with its website's content. This sitemap is as much about changes in the user's mind as it is descriptive of the site's content and navigation.

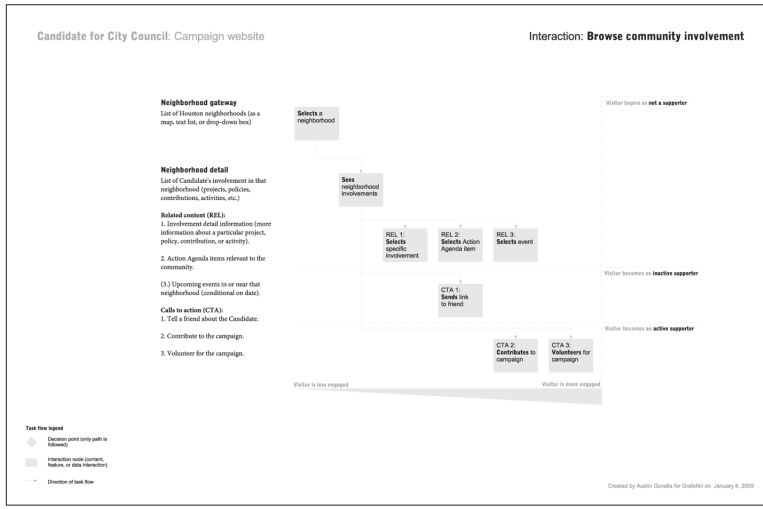

Figure 13-6. A sitemap by Austin Govella depicting growing levels of engagement in a political candidate's campaign

You'll notice that as we dug deeper, we moved from high-level sitemaps toward diagrams that isolated specific aspects of the architecture, rather than communicating the overall direction of the site. Sitemaps are incredibly flexible; while boxes and connectors can't communicate everything about a design, they are simple enough that just about anyone can both develop and understand them.

You should also note that all of these sitemaps leave out quite a bit of information. They focus on the major areas and structures of the site, ignoring many navigation elements and page-level details. These omissions are by design, not by accident. Remember the rule of thumb for sitemaps: less is more.

Keeping Sitemaps Simple

As a project moves from strategy to design to implementation, sitemaps become more utilitarian. At this stage, they are focused more on communicating the information architecture to others involved in design and development, and less on strategy and product

definition. "Lower-level" sitemaps need to be produced and modified quickly and iteratively, and often draw input from an increasing number of perspectives, ranging from visual designers to editors to programmers. Those team members need to be able to understand the architecture, so it's important to develop a simple, condensed vocabulary of objects that can be explained in a brief legend. See Figure 13-7 for an example.

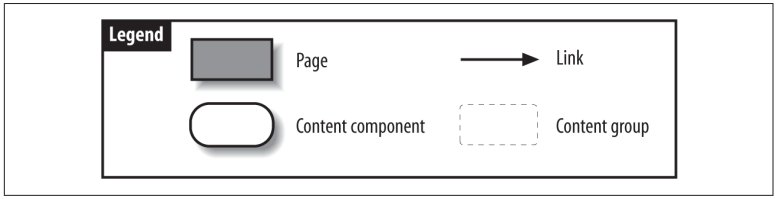

Figure 13-7. This sitemap legend describes an intentionally simple vocabulary

In this figure, the legend describes three levels of content granularity. The coarsest are content groups (made up of pages); these are followed by the pages themselves. Content components are the finest-grained content that it makes sense to represent in a sitemap. The arrow describes a link between content objects; these can be one-way or bidirectional links.

This is a minimal set of objects; we've found that retaining a limited vocabulary helps us avoid the temptation of overloading the diagram with too much information. After all, other diagrams can be used to convey other views of the architecture more effectively.

Detailed Sitemaps

As you move deeper into the implementation stage, your focus naturally shifts from external to internal. Rather than communicating high-level architectural concepts to the client, your job is now to communicate detailed organization, labeling, and navigation decisions to your colleagues on the development team. In the world of "physical" architecture, this shift can be likened to architecture versus construction. You may work closely with the client to make big-picture decisions about the layout of rooms and the location of windows; however, decisions regarding the size of nails or the routing of the plumbing typically do not involve the client. And in fact, such minutiae often need not involve the architect either.

As with physical architecture, these small details often change on the construction site: perhaps the client has changed her mind about the size of her home office, or an electrical fixture is inconveniently located in the kitchen and must be moved. In any case, change is to be expected when abstract diagrams meet the real conditions of the construction site. In our field, agile and lean development methods call for rapid iteration, often based on incomplete information. Detailed sitemaps can (and should) evolve along with the rest of the design to address the new conditions and requirements that come up during the development process in these types of projects.

That said, you should try to map out the entire environment so that the production team can implement your plans as closely as possible when starting the development process. These sitemaps must present the complete information hierarchy from the main page to the destination pages. They must also detail the labeling and navigation systems to be implemented in each area of the environment.

Sitemaps will vary from project to project, depending upon the scope. On smaller projects, the primary audience for your sitemaps may be one or two graphic designers responsible for integrating the architecture, design, and content. On larger projects, the primary audience may be a technical team responsible for integrating the architecture, design, and content through a database-driven process. Let's consider a few examples to see what sitemaps communicate and how they might vary.

Figure 13-8 shows a sitemap from the SIGGRAPH 96 Conference that introduces several concepts. By assigning a unique identification number (e.g., 2.2.5.1) to each component (e.g., pages and content chunks), the diagram presents the groundwork for an organized production process, ideally involving a database system that populates the website structure with content.

There is a distinction between a local and a remote page in Figure 13-8. A local page is a child of the main page on that sitemap, and inherits characteristics such as graphic identity and navigation elements from its parent. In this example, the Papers Committee page inherits its color scheme and navigation system from the Papers main page. On the other hand, a remote page belongs to another branch of the information hierarchy. The Session Room Layout page has a graphic identity and navigation system that are unique to the Maps area of the website.

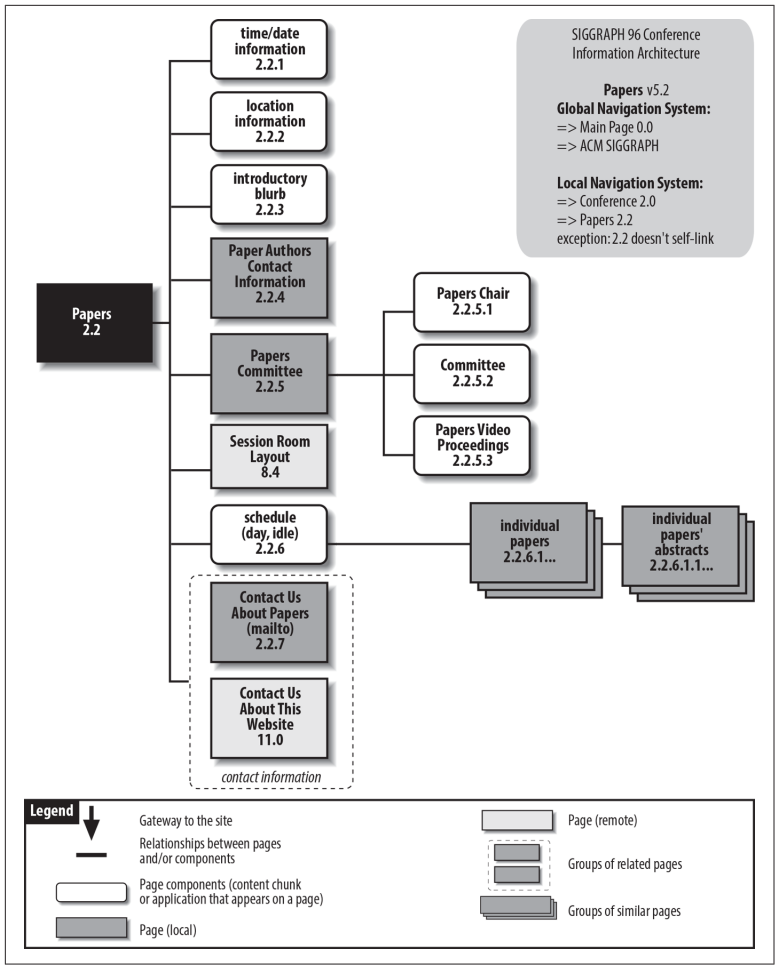

Figure 13-8. A sitemap of a major section of the SIGGRAPH conference website

Another important concept is that of content components or chunks. To meet the needs of the production process, it is often necessary to separate the content (i.e., chunks) from the container (i.e., pages). Content chunks such as "Contact Us About Papers" and "Contact Us About This Website" are sections of content composed of one or more paragraphs that can stand alone as independent packages of information. (We'll discuss content chunking in more detail later in this chapter.) The rectangle that surrounds these content chunks indicates that they are closely related. By taking this

approach, the architect provides the designer with flexibility in defining the layout. Depending upon the space each content chunk requires, the designer may choose to present all of these chunks on one page, or create a closely knit collection of pages.

You may also decide to communicate the navigation system using these detailed sitemaps. In some cases, arrows can be used to show navigation, but these can be confusing and are easily missed by the production staff. A sidebar is often the best way of communicating both global and local navigation systems, as shown in Figure 13-8. The sidebar in the upper right of this sitemap explains how the global and local navigation systems apply to this area of the website.

Organizing Your Sitemaps

As the architecture is developed, it needs to accommodate more than top-level pages. The same simple notation can be used, but how can you squeeze all of these documents onto one sheet of paper? Many applications will allow you to print on multiple sheets, but you'll find yourself spending more time taping sheets together than designing. And if a diagram is too large to print on a single sheet, it's probably also too large to reasonably view and edit on a standard monitor.

In this case, we suggest *modularizing* the sitemap. The top-level sitemap links to subsidiary sitemaps, and so on, and so on. These diagrams are tied together through a scheme of unique IDs. For example, in the top-level diagram in Figure 13-9, major pages are numbered x.0. For instance, the one representing "Committees and officers" is numbered 4.0. That page becomes the "lead page" on a new diagram (Figure 13-10), where it is also numbered 4.0. Its subsidiary pages and content components use codes starting with 4.0 in order to link them with their parent.

Using a unique identification scheme to tie together multiple diagrams helps us to somewhat mitigate the tyranny of the 8.5″ × 11″ sheet of paper, although you may still find that your architecture requires dozens of individual sheets of paper. This scheme can also be helpful for bridging a content inventory to the architectural process—content components can share the same IDs in both content inventory and sitemap. This means that in the production phase, adding content is not much different from painting by numbers.

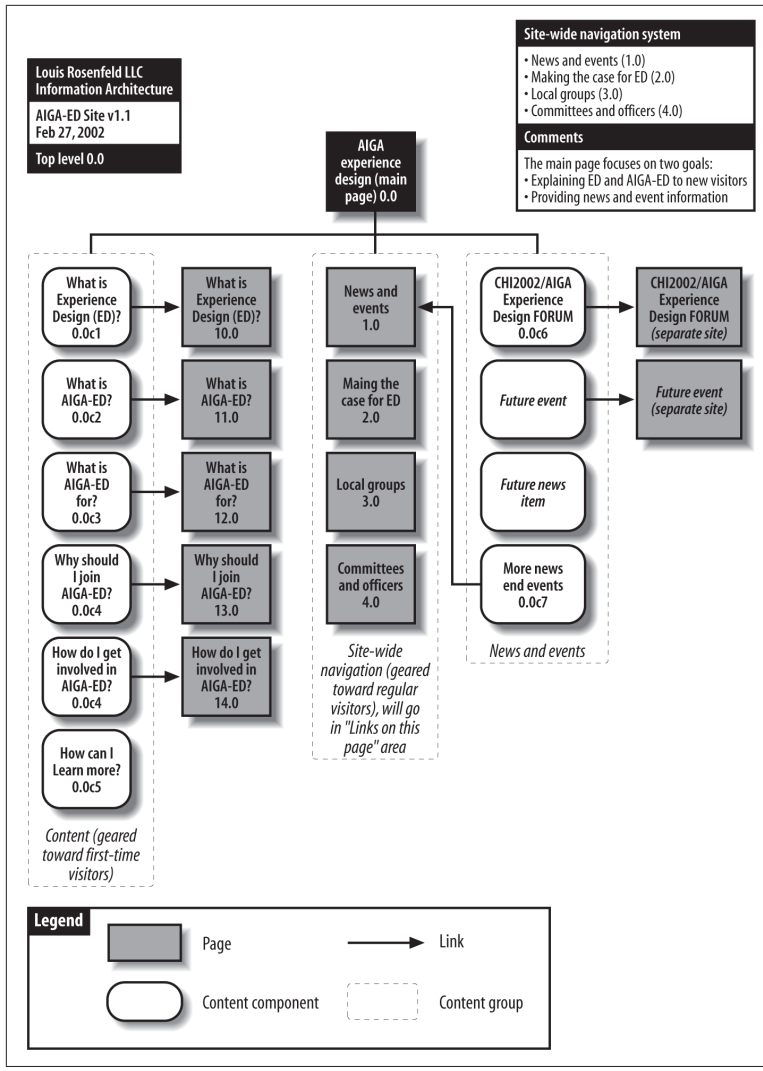

Figure 13-9. A detailed sitemap illustrating several concepts

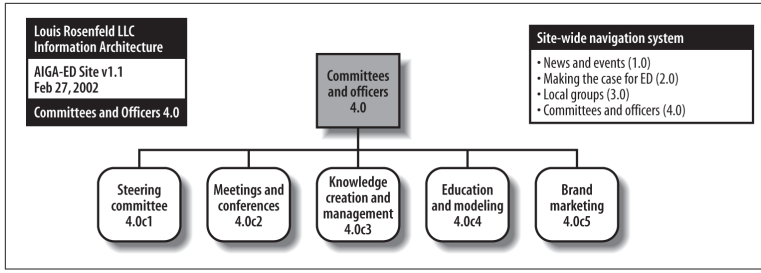

Figure 13-10. This subsidiary sitemap continues from the top-level sitemap

Wireframes

Sitemaps can help you determine where content should go and how it should be navigated within the context of a website, subsite, app, or collection of content. Wireframes serve a different role: they depict how an individual page or template should look from an architectural perspective. Wireframes connect the product's information architecture with its interaction design.

For example, the wireframe forces you to consider such issues as where the navigation systems might be located on the page or screen. And now that you see it on an early layout, does it seem that there are actually too many ways to navigate? Trying out ideas in the context of a wireframe might force you back to the sitemap's drawing board, but it's better to make such changes on paper rather than reengineering the entire system at some point in the future.

Wireframes describe the content and information architecture to be included on the relatively confined two-dimensional spaces (e.g., pages, screens). Therefore, wireframes themselves must be constrained in size. These constraints force us to make choices about what components of the architecture should be visible and accessible to users; after all, if the architectural components absorb too much screen real estate, no room will be left for actual content!

Developing wireframes also helps clarify the grouping of content components, their order, and group priority. In Figure 13-11, "Reasons to Send" is of a higher priority than the "Search Assistant." This priority is made clear by the content's prominent positioning and the use of a larger typeface for its heading.

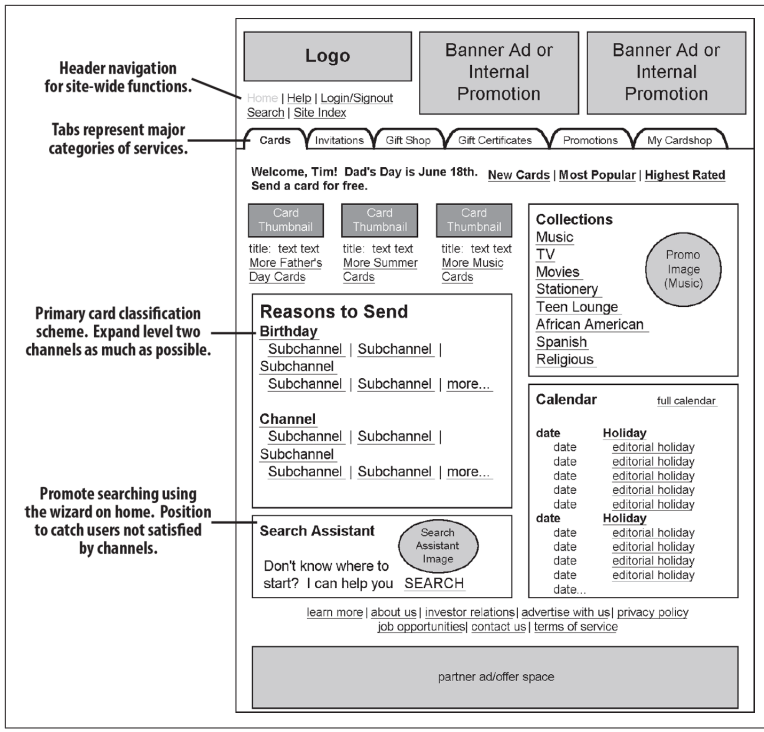

Figure 13-11. A wireframe of the main page of a greeting card site

Wireframes are typically created for the product's most important pages or screens—such as main pages, major category pages, and the interfaces to search—and other important applications. They are also used to describe templates that are consistently applied to many pages, such as content pages. And they can be used for any page that is sufficiently vexing or confusing to merit further visualization during the design process. The goal is not to create wireframes for every page or screen in your system, but only for the ones that are complicated, unique, or that set a pattern for others (i.e., templates).

Wireframes are a convenient way of exploring how page structure varies depending on screen size. Figure 13-12 shows a responsive design that accommodates reflowing for display in phone, tablet, and desktop browsers.

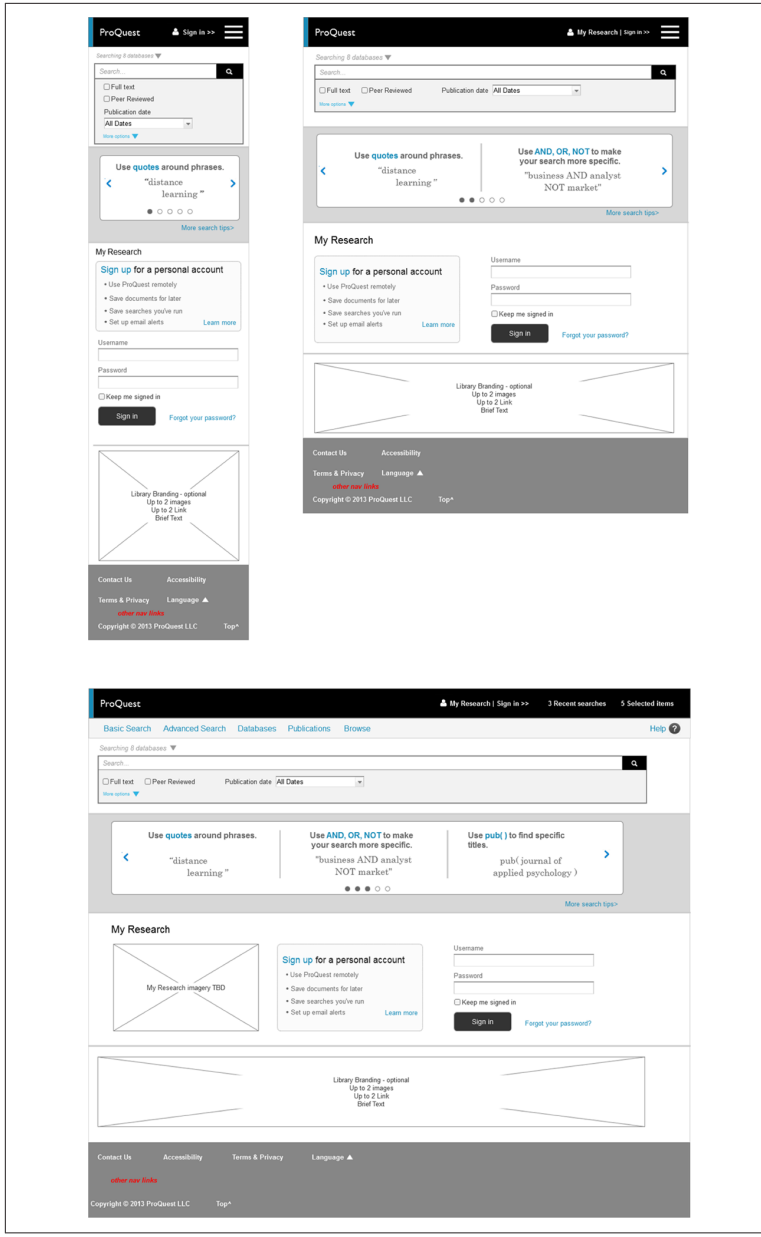

Figure 13-12. Wireframes can help designers explore the implications of varying screen sizes (wireframe developed for ProQuest LLC by Chris Farnum; reproduced with permission of ProQuest LLC—further reproduction is prohibited without permission)

Wireframes represent a degree of look and feel, and straddle the realms of visual design and interaction design. Wireframes (and page design in general) represent a frontier area where many web design–related disciplines come together and frequently clash. The fact that wireframes are produced by someone not necessarily experienced in visual or interaction design, and that they make statements about visual design (despite sometimes being rather ugly), often makes graphic designers and other visually oriented people very uncomfortable.

For this reason, we suggest that wireframes come with a prominent disclaimer that they are not replacements for "real visual design." The fonts, colors (or lack thereof), use of whitespace, and other visual characteristics of your wireframes are there only to illustrate how the site's information architecture will impact and interact with a particular page. Make it clear that you expect to collaborate with a graphic designer to improve the aesthetic nature of the overall product, or with an interaction designer to improve the functionality of the page's widgets.

We also suggest making this point verbally, while additionally conveying how your wireframe will eliminate some work that visual designers and interaction designers might consider unpleasant or not within their areas of expertise. For example, just as you'd prefer that a designer select colors or placement for a navigation bar, you've relieved the designer of the task of determining the labels that will populate that navigation bar.

Finally, because wireframes do involve visual design, their development presents a perfect opportunity for collaboration with visual designers, who will have much to add at this point. Avoid treating wireframes as something to be handed off to designers and developers, and instead use them as triggers for generating a healthy bout of interdisciplinary collaboration.

Types of Wireframes

Just like sitemaps, wireframes come in many shapes and sizes, and the level of fidelity can be varied to suit your purposes. At the low end, you may sketch quick-and-dirty wireframes on paper or a whiteboard. At the high end, wireframes may be created in HTML or with a tool like Adobe Illustrator. While the level of fidelity you use will vary depending on the stage of the development lifecycle

(with earlier stages calling for less precision), most wireframes will fall somewhere in the middle: neither too sketchy nor too precise. Let's review a few samples from the work of information architect Chris Farnum of ProQuest, a former colleague at Argus Associates and a wireframe expert. The first example (Figure 13-13) is a low-fidelity wireframe.

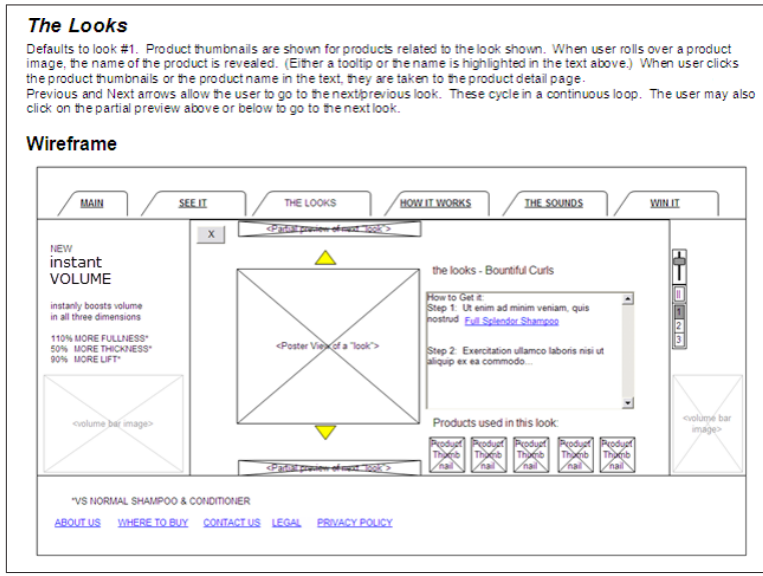

Figure 13-13. A low-fidelity wireframe; note that the focus is on layout of content and visual elements over content accuracy (wireframe developed for ProQuest LLC; reproduced with permission of ProQuest LLC—further reproduction is prohibited without permission)

Figure 13-14 shows a medium-fidelity wireframe with a high degree of detail. This wireframe was intended to introduce several aspects of content, layout, and navigation into the discussion, and was one of many wireframes used to communicate the information architecture to managers, graphic designers, and programmers.

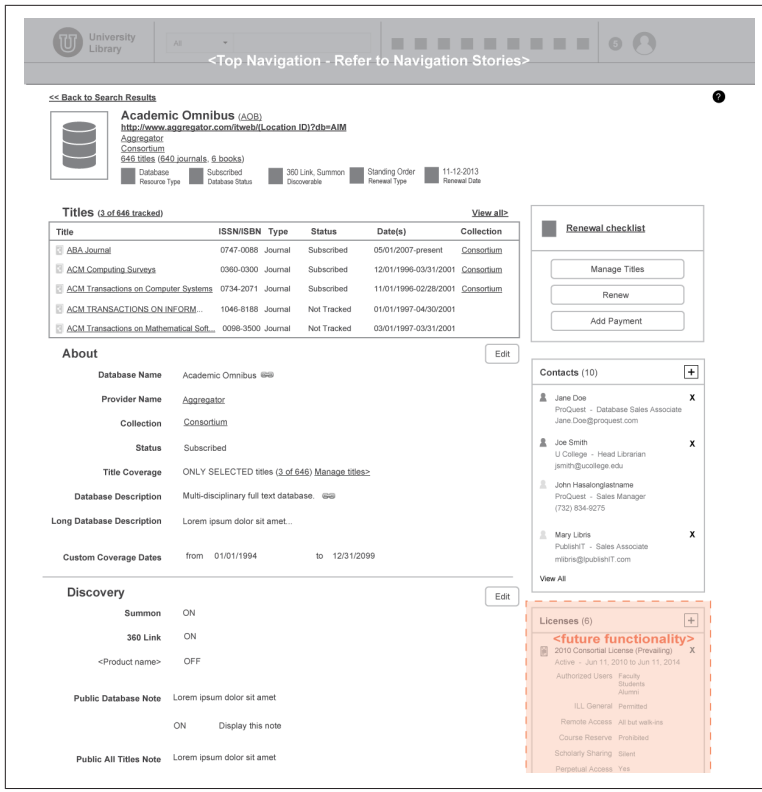

Figure 13-14. A medium-fidelity wireframe by Chris Farnum and Katherine Root; more detail, more explanation, and more unique content (wireframe developed for ProQuest LLC; reproduced with permission of ProQuest LLC—further reproduction is prohibited without permission)

Finally, Figure 13-15 is a relatively high-fidelity wireframe that presents a close approximation of what the page will actually look like. This is about as far as you can go without bringing a graphic designer into the picture.

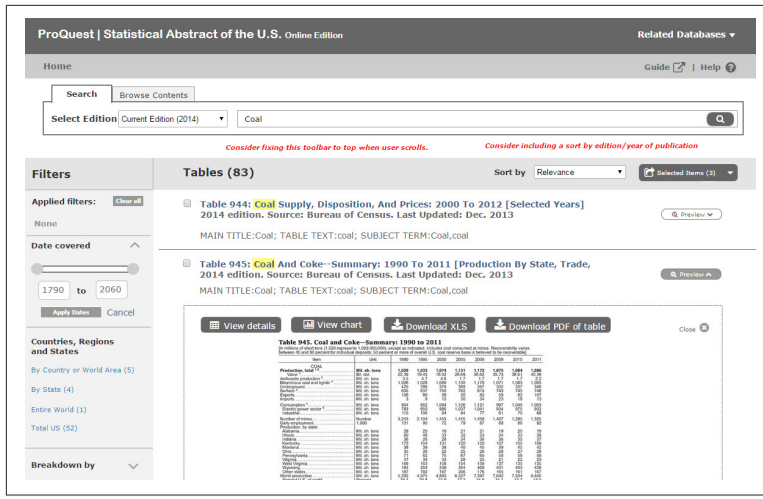

Figure 13-15. A high-fidelity wireframe (wireframe developed for ProQuest LLC by Chris Farnum; reproduced with permission of ProQuest LLC—further reproduction is prohibited without permission)

Such a high-fidelity wireframe has the following advantages:

- The content and color bring the page to life, helping to capture the attention of your clients or colleagues.
- By simulating actual page width and font size, the wireframe forces you to recognize the constraints of an HTML page.

The fidelity is sufficient to support paper prototype testing with users. On the other hand, some disadvantages are:

- Higher fidelity requires greater effort. It takes a lot of time to design such a detailed wireframe. This can slow down the process and increase costs.
- As you integrate visual elements and content into a structured layout, the focus may shift prematurely from information architecture to interface and visual design.

Provided that you recognize the strengths and weaknesses of these varying levels of fidelity, wireframes can be extremely powerful tools for communication and collaboration during the information architecture design process.

Wireframe Guidelines

Chris Farnum suggests the following best practices to consider when creating wireframes:

- Consistency is key, especially when presenting multiple wireframes. It ensures that clients will be impressed by the professionalism of your wireframes. More importantly, colleagues take wireframes quite literally, so consistency makes their design and production work go more smoothly.
- Visio and other standard charting tools support background layers, allowing you to reuse navigation bars and page layouts for multiple pages throughout the site. Similarly, Visio's stencil feature allows you to maintain a standard library of drawing objects that can be used to describe page elements.
- Callouts—small notes placed around and over your wireframes—are an effective way to provide details about the functionality of page elements. Be sure to leave room for them at the sides and top of your wireframes.
- Like any other deliverable, wireframes should be usable and professionally developed. So, tie your collection of wireframes together with page numbers, page titles, project titles, and last revision dates.
- When more than one team member is creating a project's wireframes, be sure to establish procedures for developing, sharing, and maintaining common templates and stencils (and consider establishing a wireframe "steward"). Schedule time in your project plan for synchronizing the team's wireframes to ensure consistent appearance, and for confirming that these discrete documents do indeed fit together functionally.

Content Mapping and Inventory

During research and strategy, you are focused on the top-down approach of defining an information structure that will accommodate the mission, vision, audiences, and content of the information environment. As you move into design and production, you complete the bottom-up process of collecting and analyzing the content. Content mapping is where top-down information architecture meets bottom-up.

The process of detailed content mapping involves breaking down or combining existing content into content chunks that are useful for inclusion in your environment. A content chunk isn't necessarily a sentence or a paragraph or a page. Rather, it is the most finely grained portion of content that merits or requires individual treatment.

The content, often drawn from a variety of sources and in a multitude of formats, must be mapped onto the information architecture so that it's clear what goes where during the production process. Because of differences between formats, you cannot count on a one-to-one mapping of source page to destination page; one page from a print brochure does not necessarily map onto one page on the Web. For this reason, it is important to separate content from its container at both the source and the destination. In addition, when combined with a database-driven approach to content management, the separation of content and container facilitates the reuse of content chunks across multiple pages. For example, contact information for the customer service department might be presented in context within a variety of pages throughout the system. If the contact information changes, the modification need only be made to the database record for that content chunk, and it can then be propagated through the system at the push of a button.

Even when you are creating new content, content mapping is still necessary. It often makes sense to create content in a word processing application, because tools like Microsoft Word tend to have more powerful editing, layout, and spell-checking capabilities. In such cases, you'll need to map the Word documents to HTML pages (or whatever format they are stored in in the system). The need for careful content mapping is even greater when new content is created by multiple authors throughout your organization; the mapping process then becomes an important managerial tool for tracking content from these disparate sources.

The subjective process of defining chunks should be determined by asking the following questions:

- Should this content be divided into smaller chunks that users might want to access separately?
- What is the smallest section of content that needs to be individually indexed?

- Will this content need to be repurposed across multiple documents or as part of multiple processes?

Once the content chunks have been defined, they can be mapped to their destinations or means of delivery to your audience, which can be web pages, feeds, or some other medium. You will need a systematic means of documenting the source and destination of all content so that the production team can carry out your instructions. As discussed earlier, one approach involves the assignment of a unique identification code to each content chunk.

For example, the creation of the SIGGRAPH 96 Conference website required the translation of print-based content to the online environment. In such cases, content mapping involves the specification of how chunks of content in the print materials map to pages on the website. For SIGGRAPH 96, we had to map the contents of elaborately designed brochures, announcements, and programs onto web pages. Because it wouldn't have made sense to attempt a one-to-one mapping of printed pages to web pages, we instead went through a process of content chunking and mapping with the content editor. First, we broke each page of the brochure into logical chunks of content, inventoried the results, and then devised a simple scheme tied to page numbers for labeling each chunk (Figure 13-16).

As you saw in Figure 13-9, we had already created a detailed information architecture sitemap with its own content chunk identification scheme. We then had to create a content mapping table that explained how each content chunk from the print brochure should be presented on the website (Figure 13-17).

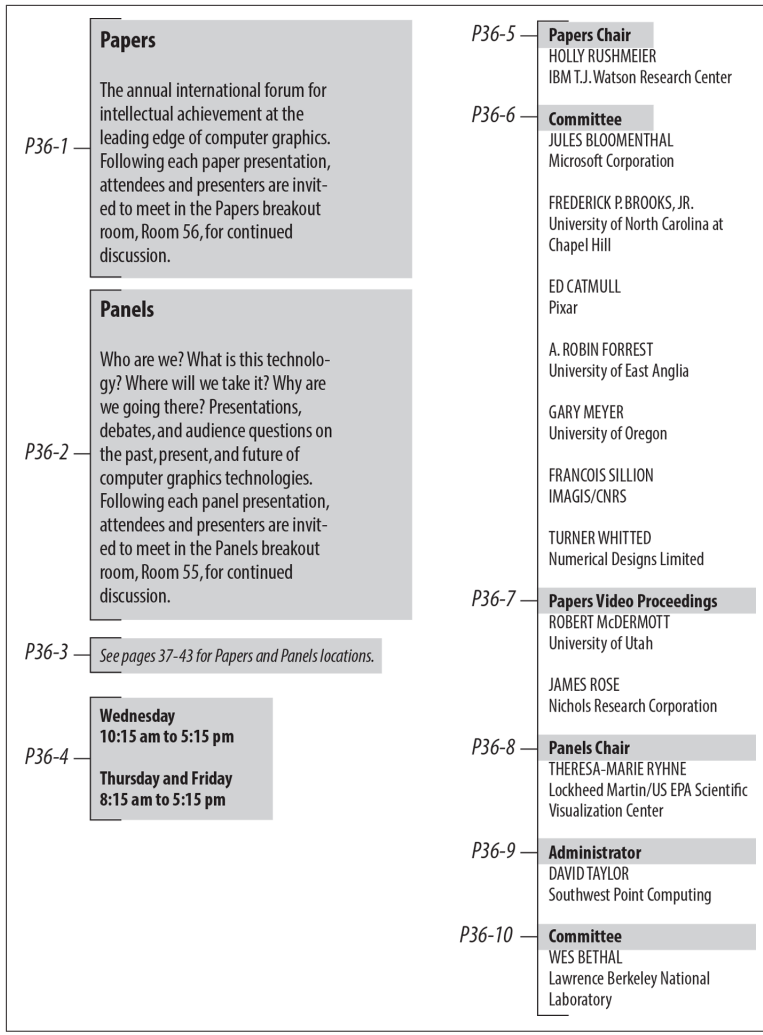

Figure 13-16. Chunks from a print brochure are tagged with unique identifiers (e.g., "P36–1") so that they can be mapped out and inventoried

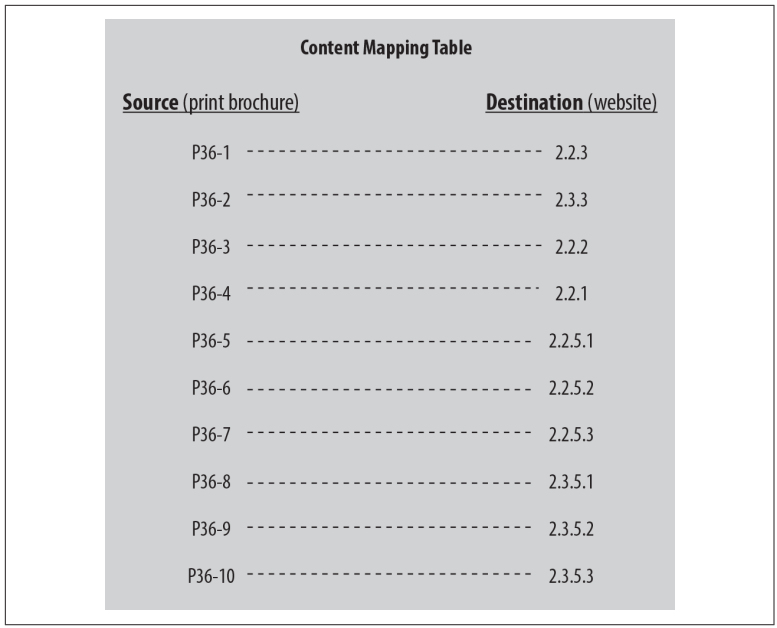

Figure 13-17. A content mapping table matches content chunks with their destinations

In this example, P36–1 is a unique ID that refers to the first content chunk on page 36 of the original print brochure. This source content chunk maps onto the destination content chunk labeled 2.2.3, which belongs in the Papers (2.2) area of the website.

Armed with the original print documents, architecture sitemaps, and the content mapping table, the production staff created and populated the SIGGRAPH 96 Conference website. As you can see in Figure 13-18, the contents of this web page (2.2) include three content chunks from P36.

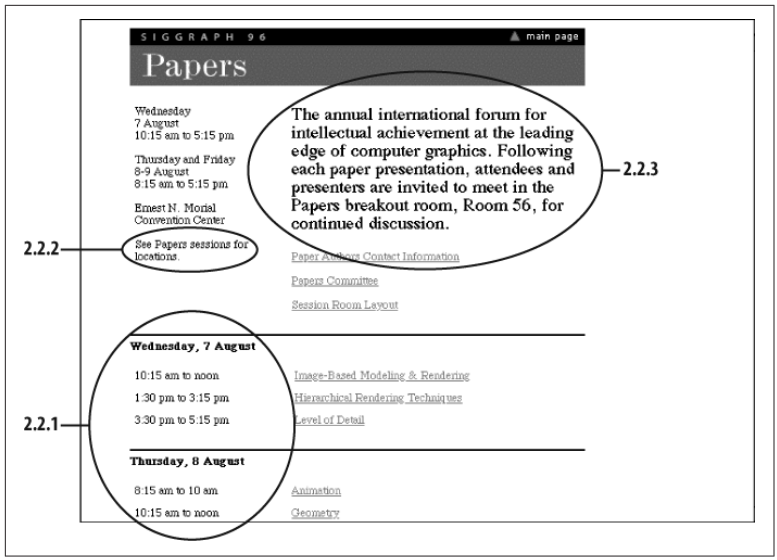

Figure 13-18. The web page produced by the content mapping process; P36–1 maps to 2.2.3, P36–3 maps to 2.2.2, and P36–4 maps to 2.2.1

Another important product of this process is a content inventory, which describes available content and where it can be found (e.g., the current site or the annual report), as well as content gaps that need to be filled. Depending upon the size and complexity of the website and the process and technologies in use for production, there are many ways to present this inventory. For larger environments, you might require a document or content management solution that leverages database technology to manage large collections of content. Many of these applications also provide a workflow that defines a team approach to page-level design and editing. For simpler systems, you might rely on a spreadsheet (see Figure 13-19). Sarah Rice of Seneb Consulting has created an excellent spreadsheet that you can download and use; in this example, she's applied it to the site of the Information Architecture Institute (formerly AIfIA).

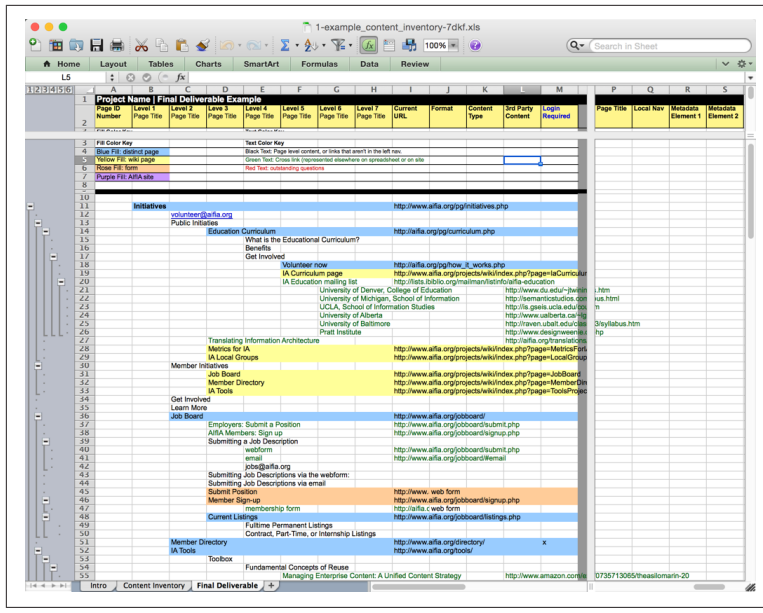

Figure 13-19. Section of a content inventory managed in Microsoft Excel

Or, if you're feeling a bit more ambitious, you can create a web-based inventory that presents the titles and unique identification numbers of each page in the site, such as that shown in Figure 13-20. Selecting the hypertext numbers pops up another browser window that shows the appropriate web page.

You can create a content inventory as soon as you have completed the content mapping process, or vice versa. And once you have an inventory of your content, you can produce a content audit: an understanding of content that needs to be created, page mockups that need to be designed, and designed pages that need to be reviewed before integration into the final product.[3]

3 For a good introduction to content inventories, see the aforementioned *Content Strategy for the Web, Second Edition,* by Kristina Halvorson and Melissa Rach (San Francisco: New Riders, 2012).

1.0	Pilot Site: Main Page
1.1	Pilot Site: Why Digital
1.2	Pilot Site: About this Pilot Program
2.0.1.A	Gateway (for subscribers)
2.0.1.B	Gateway (for non-subscribers)
2.0.2	Browser Compatibility Test
2.0.3	Browser Incompatible
2.0	Main
2.1.1	The Dissertation Abstracts Database
2.1.2	The UMI Digital Library of Dissertations
2.1.3	Future Enhancements
2.1.1.1	Submitting Electronic Theses and Dissertations
2.1.4	Feedback
2.1.5	Thank You
2.2.1	Search Results: Quick Search, Less Than 20 Hits
2.2.1.A	Search Results: Quick Search, Greater Than 20 Hits

Figure 13-20. A web-based content inventory

Content Models

Content models are information architectures made up of small chunks of interconnected content. Content models support the critical missing piece in so many information environments: contextual navigation that works deep within the product. Why is this so often a missing piece? Because it's easy—maybe too easy—for an organization to accumulate units of content, but extremely difficult to link those units together in a useful way.

Why Do They Matter?

We encounter content models all the time. A recipe is a great example. Its objects are a list of ingredients, directions, a title, and so on. If you render a recipe as "lorem ipsum,"[4] it'll still be recognizable as a recipe. But change the logic—by putting the steps before the ingredients or leaving out an important object—and the model collapses. Content models rely on consistent sets of objects and logical connections between them to work.

4 "Lorem ipsum" refers to a Latin text that is often used by designers as filler to illustrate content in presentations. For more information, see *http://en.wikipedia.org/wiki/Lorem_ipsum*.

Supporting contextual navigation

Imagine that you've found your way deep into a clothing retailer's website in a quest for a snazzy new blue oxford shirt. As a user, you've just clearly stated an incredibly specific information need. Such a need is far more precise than that of a user who has reached a site's main page. Wouldn't it be silly for the retailer not to apply this knowledge to your benefit (not to mention to its advantage)?

That's why most online retailers will, at this point, introduce you to some matching pants or other accessories. "You might also be interested in...." This is far more reasonable than a retailer expecting you to (1) guess that it sells these related items and (2) actually find those items using the top-down organization and navigation systems. Horizontal hopping across the hierarchy is a form of contextual navigation, where your movement is based more on your needs as a user than on the environment's structure. Content models exist primarily to support such navigation, whether for cross-selling retail products, connecting baseball fans to the story behind the box score, or introducing potential customers to a product's specifications.

Coping with large amounts of content

Content models also help us deal with scale. When inventorying content, it's not uncommon to stumble upon large bodies of similar information buried in our content management systems and databases. For example, after a content inventory, a company that provides information on mobile phones might find that it owns dozens of content chunks for each model's basic product information, thousands for reader reviews, and many more for information on related accessories. The phone product pages look, work, and behave the same. So do the review pages and the accessory pages.

If each type of content chunk works the same, why not take advantage of this predictability by linking them? Allow users to move naturally from a specific phone's page to its product reviews and accessories. Better yet, do this in an automated fashion so the links can be generated instantly, rather than having an army of coders deciding what should be linked to what. Automating the creation of links between content chunks means your users benefit from more and better ways to navigate contextually, and your organization derives greater value from its investment in the content.

So, content models can be especially helpful when we've got a lot of high-value content chunks that are similar to one another and aren't well linked, and some technology on hand—like your friendly neighborhood content management system—to automate the expression of those links. You can certainly create content models for smaller numbers of content chunks—for example, information associated with the dozen or so people that serve on your company's board—but it's pretty easy to manually connect these objects. You could also create content models for all of your content, but the process is a bit involved, so we recommend doing so for only your most valuable content (with value defined as a judicious combination of both user and organizational needs, of course).

An Example

Let's say you work for a media organization that has invested lots of resources in assembling information on popular music. Certain content chunks—such as artist descriptions and album pages—number in the thousands, and they all look and work in the same way. You might sense that there is potential here for a content model that serves fans of popular music. Instead of having those fans rely on the system's hierarchy to find content relevant to a particular artist or album, why not create a content model?

Based on a content inventory and audit, there are a few music-related content objects that may emerge as good candidates for a content model, shown in Figure 13-21.

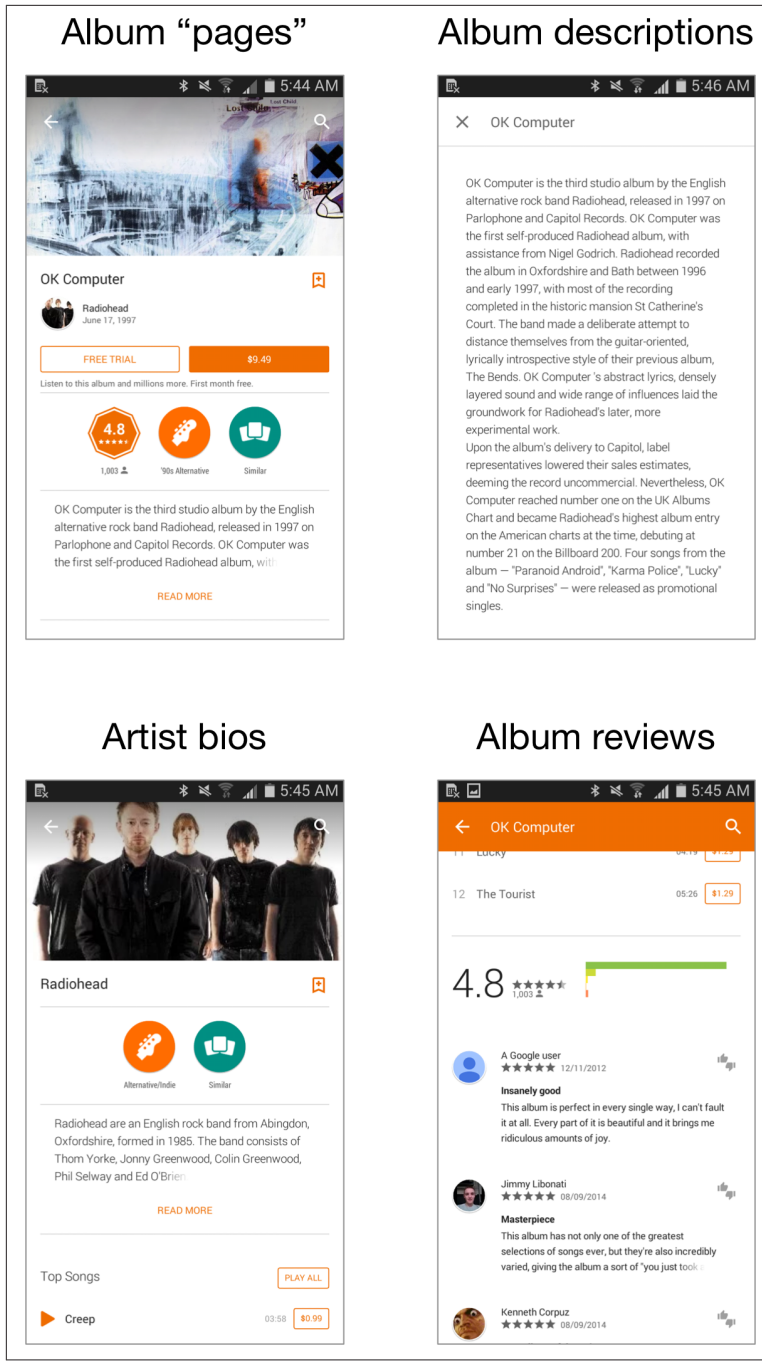

Figure 13-21. Content objects that might be the basis of a content model for album information

How should these objects be linked? We can certainly decide that an album page ought to link to its corresponding review, artist bios and descriptions should link to each other, and so on. But it won't always be so easy to come up with the most obvious links, and even if it is fairly obvious, you may need to produce some user research to validate your work.

In such cases, consider a variation of the card sort exercise. Print out a sample of each content object and cut out the navigation options (to prevent biasing users with the current information architecture). Then ask subjects to look at each content object and consider where they'd want to go next. Then have them cluster the objects and draw lines between them that indicate navigation (they can do this with string, or they can tape the content object samples to a whiteboard and use dry-erase markers to draw their lines). Arrows indicate whether users wish to navigate in both directions or prefer a one-way link.

To perform a simple gap analysis, ask subjects which missing content objects would be nice to include in the mix. By doing so, you'll get a sense of what should be added to your content model. If you're fortunate, the missing objects might already exist somewhere else in your site. Otherwise, you'll at least have some guidance in deciding which content to create or license.

At the end of the process—whether based on user research or your own hunches—you'll have an idea of how your content model ought to work. The result might look like Figure 13-22.

You've now identified new content objects, like a discography, that you might need to create. And you've linked to other content, like YouTube videos of the band and events in a concert calendar, that is a logical extension of the content model (and possibly, a connection to candidates for future content models). You've also identified logical "tops" or common points of entry to this content. And ultimately, you have a sense of how users might want to navigate an area deep in the guts of your site.

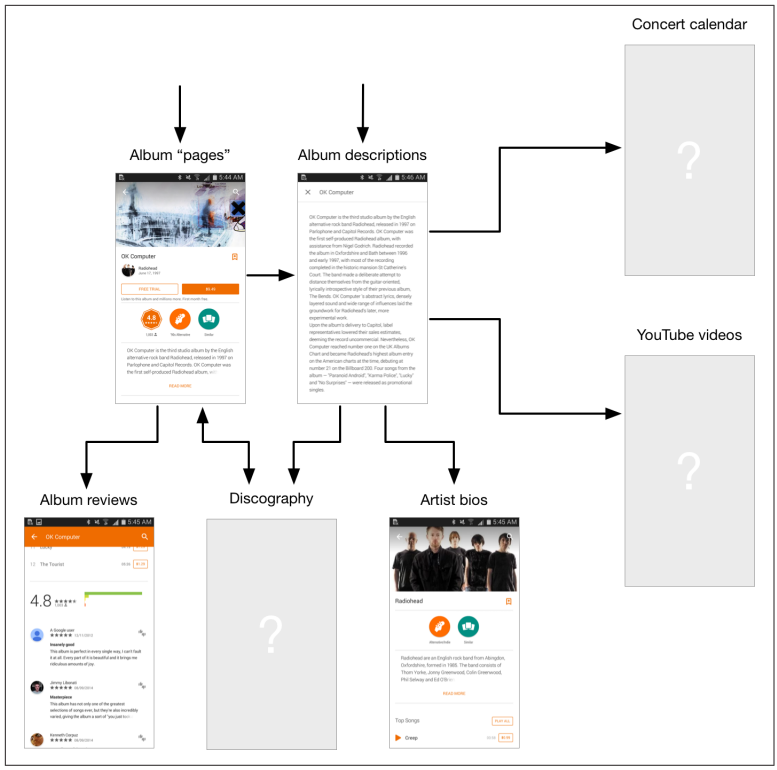

Figure 13-22. An ideal content model, showing navigation and missing content objects

Unfortunately, you're not quite done. How do these links between content objects get made?

If you're Amazon, you've got reams of usage data to draw from. Amazon employs customer behavior data to make connections between related products in its content model; familiar examples are the products listed under "Customers who bought this item also bought" and "What do customers ultimately buy after viewing this item?" But not every organization has the traffic volume from which to cull this kind of useful data.

So, the rest of us typically rely on metadata as the basis of the logic that connects our content chunks. Shared metadata does the work of linking a pair of content chunks. For example, if we want to link an album page and an album review, the logic might look like this:

```
IF ALBUM PAGE'S ALBUM NAME = ALBUM REVIEW'S ALBUM NAME
THEN LINK ALBUM PAGE AND ALBUM REVIEW
```

Now, this rule might suffice for albums with unique titles, like *OK Computer*. But what if the title is the ubiquitous *Greatest Hits*? If you're lucky, the object has a unique identifier, like an ISBN, that can be used as connecting metadata:

```
IF ALBUM PAGE'S UNIQUE ID = ALBUM REVIEW'S UNIQUE ID
THEN LINK ALBUM PAGE AND ALBUM REVIEW
```

But as that's often not the case, your linking logic will likely need to get a little more complicated, and additional metadata attributes will be necessary:

```
IF ALBUM PAGE'S ALBUM NAME = ALBUM REVIEW'S ALBUM NAME
AND ALBUM PAGE'S ARTIST NAME = ALBUM REVIEW'S ARTIST NAME
THEN LINK ALBUM PAGE AND ALBUM REVIEW
```

As you can see, these rules rely on metadata. Do the required metadata attributes exist? The bad news is that you'll probably need to invest in creating new metadata from scratch (or acquiring it).

Of course, metadata availability is a consideration with just about any information architecture project of any size. And the good news is that the content modeling process will help you decide which metadata attributes to invest in by helping you select the most useful from the wide range of possibilities.

Consider our arrows in Figure 13-22. Which metadata will be necessary to drive the logic behind each link? You can make a simple table listing each content object, which other objects it should link to, and the metadata attributes required to make those connections. It might look something like Table 13-2.

Table 13-2. Content object linking table

Content objects...	link to other content objects...	by leveraging common metadata attributes
Album page	Album review, discography, artist	Album Name, Artist Name, Label, Release Date
Album review	album page	Album Name, Artist Name, Review Author, Source, Pub Date
Discography	Album review, artist description	Artist Name, Album Name, Release Date
Album description	Artist bio, discography, concert calendar, TV listing	Artist Name, Desc Author, Desc Date

Content objects...	link to other content objects...	by leveraging common metadata attributes
Artist bio	Artist description	Artist Name, Individual Artist Name
Concert calendar	Artist description	Artist Name, Tour, Venue, Date, Time
YouTube listing	Artist description	Video Title, URL, Amount of Views

Notice a pattern here? Certain metadata attributes show up more frequently than others. These are the attributes that are most necessary for the content model to succeed. If you're operating with limited resources (and who isn't?), now you'll have an excellent way to prioritize your investment in metadata attributes.

A Valuable Process

As you can see, content models are as much an exercise as a deliverable. While the primary output is a useful IA deliverable that informs the design of contextual navigation deep within an information environment, the process also generates two invaluable, if secondary, benefits.

First, content modeling forces us to determine which content is the most important content to model. As you can see, it's work. Most likely you can't create content models for all of your content. So you'll have to ask yourself: which content fulfills the requirements of homogeneity, high volume, and, most of all, high value? You might find a set of priorities falls out of this exercise; for example, perhaps this year you'll develop a product area content model, next year a support area content model, and later you'll link those two models together for even greater benefit.

Second, content modeling also forces you to choose which of the many metadata attributes are the ones that will make your content model operational. The combination of focusing on and narrowing down to critical content and critical metadata means a huge simplification and clarification of a large and complex problem space.

Controlled Vocabularies

There are two primary types of work products associated with the development of controlled vocabularies. First, you'll need metadata matrices that facilitate discussion about the prioritization of

vocabularies (see Table 13-3 for an example). Second, you'll need a tool to manage the vocabulary terms and relationships.

Table 13-3. A metadata matrix for 3Com

Vocabulary	Description	Examples	Maintenance
Subject	Terms that describe networking	Home networking; servers	Difficult
Product type	Types of products that 3Com sells	Hubs; modems	Moderate
Product name	Names of products that 3Com sells	PC Digital WebCam	Difficult
Product brand	Brands of products that 3Com sells	HomeConnect; SuperStack	Easy
Technology	Types of technologies associated with products	ISDN; broadband; frame relay	Moderate
Protocols	Types of standards and protocols associated with products	TCP/IP; Ethernet	Moderate
Hardware	Types of devices that products are used in	PDA; wireless phone; Internet appliances; PC	Moderate
Geographic location: region	Name of geographic region	Europe; APR	Easy
Geographic location: country	Name of country	Germany; Czech Republic	Easy
Language	Name of language	German; Czech	Easy
Technology applications	Names of applications for technologies	Call center; ebusiness	Moderate
Industries	Types of industries that 3Com works with	Healthcare; government	Easy
Audiences	Kinds of audiences the 3Com site attracts	Consumers; first-time visitors; media	Easy
Customer group: workplace	Type of workplace that customers work in	Home; office	Moderate
Customer group: business	Size or scale of business that customers work in	Small business; large enterprise; service provider	Moderate
Roles	Type of role that people have in their business	IT manager; consultant	Moderate
Document type	Purpose of content object	Form; instructions; guide	Easy

As you can see from Table 13-3, there's no shortage of possible vocabularies. Your job is to help define which vocabularies should be developed, considering priorities and time and budget

constraints. A metadata matrix can help you to walk clients and colleagues through the difficult decision-making process, weighing the value of each vocabulary to the user experience against the costs of development and administration.

As you shift gears from selecting vocabularies to building them, you'll need to choose a database solution to manage the terms and term relationships. If you're creating a sophisticated thesaurus with equivalence, hierarchical, and associative relationships, you should seriously consider investing in thesaurus management software (see Chapter 10 for further discussion). However, if you're creating a simple vocabulary with only preferred and variant terms, you should be able to manage with just a word processor, spreadsheet program, or basic database.

When we created a controlled vocabulary to be used by thousands of representatives at AT&T's inbound call centers, we managed the accepted and variant terms in Microsoft Word (see Table 13-4).

Table 13-4. Excerpt from a controlled vocabulary database created for AT&T

Unique ID	Accepted term	Product code	Variant terms
PS0135	Access Dialing	PCA358	10–288; 10–322; dial around
PS0006	Air Miles	PCS932	AirMiles
PS0151	XYZ Direct	DCW004	USADirect; XYZ USA Direct; XYZDirect card

For this project, we were dealing with 7 distinct vocabularies and around 600 accepted terms:

- Products & Services (151 accepted terms)
- Partners & Competitors (122 accepted terms)
- Plans & Promotions (173 accepted terms)
- Geographic Codes (51 accepted terms)
- Adjustment Codes (36 accepted terms)
- Corporate Terminology (70 accepted terms)
- Time Codes (12 accepted terms)

Even given the relatively small size and simplicity of these vocabularies, we found Microsoft Word was barely sufficient for the task. We created one very long document with tables for each vocabulary.

This document was "owned" by a single controlled vocabulary manager and shared via our local area network. Our team of indexing specialists were able to search against accepted and variant terms in the "database" using MS Word's Find capability. And we were able to output tab-delimited files to assist the programmers who were building the site at AT&T.

Design Collaboration

Once you've developed sitemaps, wireframes, content models, and vocabularies, you'll find yourself collaborating more with other people involved in developing the product—visual designers, developers, content authors, or managers. You'll move from capturing and communicating your own design concepts to integrating them with the visions of other members of your team. Naturally, this is as challenging as design gets—everyone wants his own ideas to play a role in the final product, and because the group's members often come from interdisciplinary backgrounds, there are often competing vocabularies and breakdowns in communication. But if each person goes in with an open mind and good tools for collaborating, this difficult phase is also the most gratifying one, ending with a shared vision that's far better than anyone was likely to arrive at individually. Design sketches and web prototypes are just two tools for merging differing ideas.

Design Sketches

In the research phase, the design team developed a sense of the desired graphic identity or look and feel. The technical team assessed the information technology infrastructure of the organization and the platform limitations of the intended audiences, and they understood what was possible with respect to features such as dynamic content management and interactivity. And, of course, the architect designed the high-level information structure for the environment. Design sketches are a great way to pool the collective knowledge of these three teams in a first attempt at interface design for the top-level pages of the website or app. This is a wonderful opportunity for interdisciplinary user interface design.

Using the wireframes as a guide, the designer now begins sketching pages of the product on sheets of paper. As the designer sketches

each screen, questions arise that must be discussed. Here is a sample sketching-session dialog:

> Developer: "I like what you're doing with the layout of the main screen, but I'd like to do something more interesting with the navigation system."
>
> Designer: "Can we implement the navigation system using pull-down menus? Does that make sense architecturally?"
>
> You: "That might work, but it would be difficult to show context in the hierarchy. How about a table of contents? We've had pretty good reactions to that type of approach from users in the past."
>
> Developer: "We can certainly go with that approach from a purely technical perspective. How would a tear-away table of contents look? Can you sketch it for us? I'd like to do a quick-and-dirty prototype."

As you can see, the design of these sketches requires the involvement of members from each team. It is much cheaper and easier for the group to work with the designer on these rough sketches than to begin with code and finished graphics. These sketches allow rapid iteration and intense collaboration. The final product of a sketching session might look something like Figure 13-23.

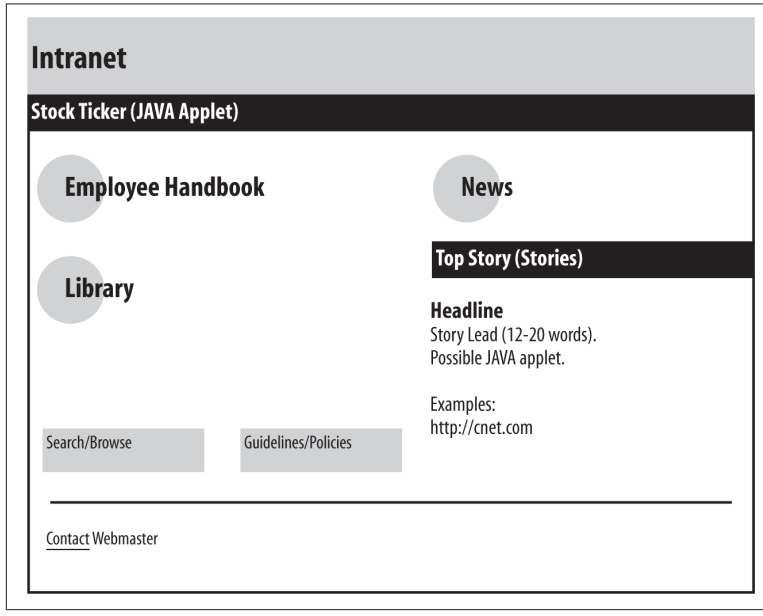

Figure 13-23. A basic design sketch

In this example, Employee Handbook, Library, and News are grouped together as the major areas of the website. Search/Browse and Guidelines/Policies make up the page navigation bar. The News area defines space for a dynamic news panel. This sketch may not look much different from a wireframe. In fact, the team may have begun with a wireframe, then iterated on the design until arriving at this sketch, which in turn may be the basis for a revised and final wireframe.

Starting with a sketch—whether a formal wireframe or something more "back-of-the-napkin"—is critical to the success of interdisciplinary meetings.

The sketch provides a common focus for each participant, minimizing the attention paid to the individual personalities around the table. It also makes it more likely that participants will be using the same terminology to discuss the design; shared terms for design concepts often emerge directly from the sketch itself.

Finally, note that design sketches aren't necessarily "owned" by the team responsible for the information architecture. For example, sketches that describe functional requirements may be under the purview of the designer or developer. Be wary of getting caught up in ownership issues; contributing to the design, regardless of who is driving Visio, OmniGraffle, or Illustrator, is far more important to the project's outcome.

Interactive Prototypes

A high point of the design process is the creation of interactive prototypes.[5] More than sketches or scenarios, these digital renditions show how the product will look and function. They are concrete and often aesthetically compelling; you can actually see how your work will really come together, and maybe even kick the tires yourself.

While the balance of attention now shifts toward aesthetic considerations such as page layout and graphic identity, the prototypes

5 For more on the creation of prototypes, see Todd Zaki Warfel's *Prototyping: A Practitioner's Guide* (Brooklyn, NY: Rosenfeld Media, 2011).

frequently identify previously unseen problems or opportunities related to the information architecture. Once your architecture and navigation system are embodied in an actual interactive system, it becomes much easier for you and your colleagues to see whether they are working.

The designer may begin with two concepts based on a single information architecture. After getting feedback from the client, the design team may work together to adapt and extend the preferred concept. At this point, conceptual design officially ends, and production actually begins. The most exciting challenges for the architecture have been met, and you now begin the days of detail.

Point-of-Production Information Architecture

Ideally, the production process would proceed smoothly in a paint-by-numbers manner, and you could sit back and relax. In reality, you must be actively involved to make sure the architecture is implemented according to plan and to address any problems that arise. After all, you can't anticipate everything.

Many decisions must be made during production. Are these content chunks small enough that we can group them together on one page, or should they remain on separate pages? Should we add local navigation to this section of the environment? Can we shorten the label of this page? Be aware that at this stage, the answers to these questions may impact the burden on the production team as well as the usability of the product. You need to balance the requests of your client against the sanity of the production team, the budget and timeline, and your vision for the information architecture of the environment.

You shouldn't need to make major decisions about the architecture during production, because hopefully these have already been made. Discovering a major flaw in the architecture at this point is a nightmare. Fortunately, if you've followed the process of research, strategy, and design, this is unlikely. You have worked hard to define the mission, vision, audiences, and content for the product. You have documented the decisions made along the way. You have resolved the top-down and bottom-up approaches through content mapping and detailed sitemaps. Through careful planning, you've created a solid information architecture that should stand the test of time.

Still, it's worth reminding yourself that an information architecture can never be perfect. Factors of content, people, and context are constantly changing, and the architecture will, too. It's more important to invest your energy in educating your colleagues that information architecture design is an ongoing process, rather than fighting with them to get it "right."

Putting It All Together: Information Architecture Style Guides

Information environments are always growing and changing. You must help guide their development—even after the product launches—or risk architectural drift, or worse: a decaying user experience that doesn't evolve with its users. It's frustrating to see your carefully and flexibly designed organization, navigation, labeling, and indexing systems get mangled as maintainers add content without heeding the architectural implications. While it may be impossible to completely prevent the effects of entropy, an *information architecture style guide* can steer content maintainers in the right direction.

An architecture style guide is a document that explains how the environment is organized, why it is organized that way, who it's for, and how the architecture should be extended as the system grows. You should begin your guide with documentation of the mission and vision for the product, as it's important to understand the original goals. Continue with information about the intended audiences. Who was it designed for? What are their goals? What assumptions were made about their information needs? Then, follow up with a description of the content development policy. What types of content will and won't be included, and why? How often will it be updated? When will it be removed? And who will be responsible for it?

The "Why" Stuff

Documenting the lessons learned and the decisions made during the research, strategy, and design phases is critical. These underlying philosophies not only drive the design and maintenance of the information architecture, but also guide your product through the zigs and zags of major changes that your organization will surely encounter in the future.

For example, your organization may merge with another or spin off a unit. It may offer new products, or try to reach new markets and go global in the process. Major changes like these often coincide with major organizational changes such as the appointment of new senior managers, many of whom wish to leave their mark in all areas, including the product's design. But do new requirements and major changes to the organization require major changes to the site's information architecture? Ideally, not; a clearly documented rationale explains an information architecture and demonstrates its flexibility, which mitigates against the extremes that plague so many redesigns.

Perhaps the biggest "why" you'll encounter is the one that comes so often from senior vice presidents, marketing managers, and product managers, which in effect boils down to "Why can't my favorite feature/my department's content be made more prominent/become your highest priority?" An information architecture style guide provides you with concrete documentation to help you prioritize the many such requests you'll likely encounter. It'll even provide you with cover when you absolutely have to say no.

The "How" Stuff

Your style guide should include some basic nuts-and-bolts components to help various people maintain the environment. Consider including such sections as:

Standards
: There are usually at least a few rules that must be followed while maintaining and changing the environment. For example, newly created documents must be indexed with terms from the appropriate controlled vocabulary before they are published. Or there may be specific procedures that must be followed to ensure that new content is immediately crawled and indexed by the search system. Here's the place to note the rules...

Guidelines
: ...and distinguish the rules from the guidelines, which suggest—but don't mandate—how the information architecture should be maintained. These may be drawn from information architecture

best practices,[6] and often require interpretation for each situation in which you'll find yourself; examples include advice on how to avoid overly long lists of links and page-titling recommendations.

Maintenance procedures
: Regular tasks that are required for the environment's survival should be fully documented, such as when and how to add new terms to a controlled vocabulary.

Pattern library
: Consider creating a pattern library[7] that documents and provides access to reusable aspects of your product's design—such as a navigation widget that helps users scroll through pages of results—to cut down on reinventing the wheel.

Your style guide should also present the sitemaps, wireframes, controlled vocabulary information, and other documentation that came from the design process and will be reused throughout the environment's lifetime. Because you won't always be there to explain these deliverables, it may be necessary to provide written explanations to accompany the sitemaps. You also need to create guidelines for adding content to ensure the continued integrity of the organization, labeling, navigation, and indexing systems. This can be a challenge. When should a new level in the hierarchy be added? Under what conditions can new indexing terms be introduced? How should local navigation systems be extended as the website grows? By thinking ahead and documenting decisions, you can provide much-needed guidance—a user's manual, really—to the environment's maintainers.

6 For a few examples of IA heuristics, check out Lou's "IA heuristics" and "IA heuristics for search systems".

7 To learn how Yahoo! developed its excellent library, read "Implementing a Pattern Library in the Real World: A Yahoo! Case Study", by Erin Malone, Matt Leacock, and Chanel Wheeler.

Keep in mind the different audiences that might use the style guide. For example, in a large organization, content authors working from far-flung parts of the globe may not need to know the site's overall strategy so much as the maximum number of characters they should use for a document title. Conversely, designers may need to understand the rules that guide construction of the `alt` text that a navigation system's mouseovers rely upon. Consider an information architecture style guide as a sort of "how and why" document that should be designed for use, just like any other information system. And remember that your organization may already have a style guide for its branding, its content, and other aspects of its online presence; when possible, integrate information architecture guidelines into existing style guides.

Recap

OK, let's recap what we learned in this chapter:

- In the design phase, the emphasis of the project moves from process to deliverables—it's where the information architecture starts to become manifest.
- That said, these deliverables aren't the whole story—process is as important during this phase as it is during research and strategy.
- Information architectures are abstract and conceptual, which makes it difficult to capture them in diagrams.
- You should provide multiple "views" of your information architecture to display its different aspects.
- These views should be developed for specific audiences and needs.
- IA diagrams define content components and the connections between them.
- Sitemaps show the relationships between information elements such as pages and other content components, and can be used to portray organization, navigation, and labeling systems.
- Wireframes depict how an individual page or template should look from an architectural perspective.
- Content models support contextual navigation that works deep within the product.

- Controlled vocabularies can be conveyed with metadata matrices and applications that enable the vocabulary to be managed.
- As you move through the design phase, you'll find yourself collaborating more with other people involved in developing the product—an open mind and good collaboration tools are essential.