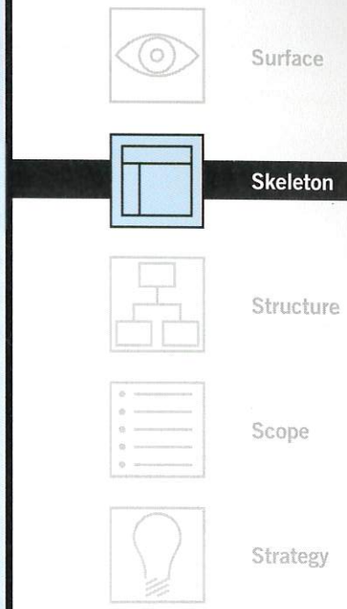
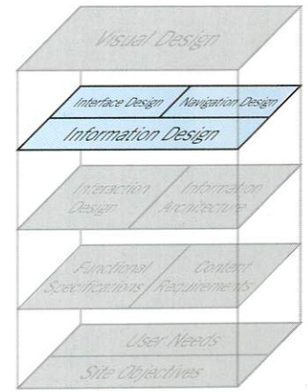
chapter 6

# The Skeleton Plane

## Interface Design, Navigation Design, and Information Design

The conceptual structure begins to give shape to the mass of requirements arising from our strategic objectives. On the skeleton plane, we further refine that structure, identifying specific aspects of interface, navigation, and information design that will make the intangible structure concrete.

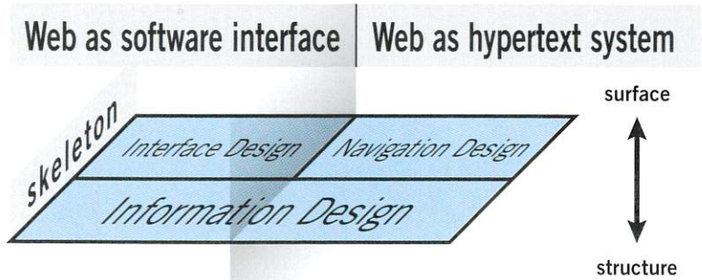


## Defining the Skeleton

The structure plane covered in the preceding chapter defines how our site will work; the skeleton plane defines what form that functionality will take. In addition to addressing more concrete issues of presentation, the skeleton plane also deals with matters that involve a more refined level of detail. On the structure plane, we looked at the large-scale issues of architecture and interaction; on the skeleton plane, our concerns exist almost exclusively at the level of individual pages and their components.

On the software side, we define the skeleton through **interface design**—the familiar realm of buttons, fields, and other interface components. But information spaces have a unique set of problems all their own. **Navigation design** is the specialized form of interface design tailored to presenting information spaces. Finally, crossing both sides, we have **information design**, the presentation of information for effective communication.






These three elements are closely bound together—more so than any of the other elements covered in this book. It's not uncommon to be faced with interface design problems that begin to blur into information design problems, or to encounter questions about information design that turn out to be matters of navigation design.

Even though the lines sometimes get blurry, identifying these as separate areas of concern helps us better assess whether we've settled on a suitable solution. Good navigation design can't correct bad information design; if we can't tell the difference between the types of problems, we can't tell if we've really solved them.

If it involves providing users with the ability to *do things*, it's interface design. The interface is the means by which users actually come into contact with the functionality defined in the specifications and structured in the interaction design.

If it involves providing users with the ability to *go places*, it's navigation design. The information architecture applied a structure to the list of content requirements we developed; the navigation design is the lens through which the user can see that structure, and is the means by which the user can move through it.

If it involves *communicating ideas* to the user, it's information design. This is the broadest of the three elements on this plane, potentially incorporating or drawing upon aspects of almost everything we've seen so far on both the software interface side and the hypertext information system side. Information design crosses the boundary between task-oriented software systems and information-oriented hypertext systems because neither interface design nor navigation design can be fully successful without good information design to support them.

## Convention and Metaphor

Habit and reflex are the foundation for much of our interaction with the world—indeed, if we weren't able to reduce so much of what we do to reflex, we'd accomplish a whole lot less every day. Can you imagine if driving a car never got any easier than it was the first time you tried it? Your ability to drive, cook a meal, or use a computer—without being thoroughly exhausted by the tremendous concentration needed for the task—depends on the accumulation of lots of tiny reflexes.

Convention allows us to apply those reflexes in different circumstances. I used to have a car that invariably caused trouble whenever any of my friends drove it. When they started the car, the first thing they did was wash the windshield. This wasn't because they thought the windshield was dirty (though it probably was); rather, it was because they were trying to turn on the headlights. The controls on my car were different from the conventions they were used to.

Telephones are another good example of the importance of convention. From time to time, manufacturers have experimented with deviations from the standard three-by-four layout for the buttons on a telephone, such as two rows of six buttons each, or three rows of four. Circular arrangements still pop up from time to time, but these are becoming ever more rare as the rotary-dial phones on which they are based fade into the mists of technological oblivion.



It seems like the layout shouldn't make that much of a difference, but it does. If you measured the time a user spends trying to figure out which button to push on a nonstandard telephone, it might turn out to be something like three seconds per call. Not that big a difference—but to the user, those three seconds aren't just lost time. Those three seconds are filled with pure frustration, as a reflexive task becomes agonizingly slow simply because the rug of convention has been pulled out from under the user's feet.

In fact, the telephone's three-by-four matrix of digits is so well ingrained that it has become the standard for other devices that have nothing to do with telephones, such as microwave ovens or remote controls for televisions and VCRs. (Interestingly, the phone pad is not the only standard in this area: The "10-key" standard used by old adding machines, which inverts the order of the digits on the telephone keypad, can now be seen on calculators, computer keyboards, ATMs, cash registers, and in specialized data-entry applications such as inventory systems. Because both standards use a three-by-four matrix, people can adapt to either with relative ease, though a single standard would really be the best solution of all.)

This is not to say that the answer to every interface problem is slavish adherence to convention. Instead, you should simply be cautious about deviating from convention and only do so when a different approach offers clear benefits. Creating a successful user experience requires having explicitly defined reasons for every choice that you make.

Making your interface consistent with others that your users are already familiar with is important, but even more important is making your interface consistent with itself. The conceptual models for the features of your site can help you ensure internal consistency. If you have two features that both have the same conceptual model, they are likely to have similar interface requirements. Using the same conventions in both places allows a user who is familiar with one to adapt quickly to the other.

Even where the conceptual models for features differ, ideas that apply across a variety of conceptual models should be treated similarly (if not identically) wherever they appear. Concepts like "start," "finish," "go back," or "save" can be found in a wide range of contexts. Giving these a consistent treatment throughout lets users apply what they already know from having used other parts of the system, getting them to their goals faster and with fewer mistakes.

Just as you shouldn't take the conceptual models underlying your interaction design too literally, you should resist the impulse to construct your site around a series of **metaphors**. Metaphors for the features of your site are cute and fun, but they almost never work as well as it seems they should. In fact, they often don't work at all.

In some cases, you might be inclined to pattern the interface design for a particular function after the interface of a real-world object. Remember Slate's navigation in which you could "turn" the pages just like in a real magazine? Most interfaces and navigational devices in the real world are the product of the constraints of the real world: physics, the properties of materials, and so on. The Web has few of the same constraints.



Drawing analogies between features of your site and experiences people have in the real world might seem like a good way to help people get a handle on what those features are all about. However, this kind of approach usually obscures the nature of the feature instead of revealing it. Even though the association between the feature and its metaphorical representation is clear to you, it's just one of any number of associations your users might apply—especially if those users come from a different cultural background than you do. What does that little picture of a telephone mean? Can I make a phone call from this site? Check my voice mail? Pay my phone bill?

Of course, the content of your site should provide some degree of context to help users make better guesses about what features your metaphors are intended to represent. But the more diverse the range of content and functionality you offer, the less reliable these guesses become—and at any rate, some part of your audience is always going to guess incorrectly. It would be better (and simpler) to eliminate the user's guesswork altogether.

Avoiding metaphors is really just about reducing the mental effort required for users to get around and use the functionality of your site. Having an icon of a phone book to represent an actual directory of telephone numbers might be just fine; having a picture of a coffee shop to represent your chat area is a bit more problematic.

## Interface Design

Successful interfaces are those in which users immediately notice the important stuff. Unimportant stuff, on the other hand, doesn't get noticed—sometimes because it's not there at all. One of the biggest challenges of designing interfaces for complex systems is figuring out which aspects the users don't need to deal with and reducing their visibility (or leaving them out altogether).

For people with a background in programming, this way of thinking can require some adjustment. It's often very different from what they are used to. Good programmers always take into account the most unlikely scenarios (called "edgecases" in programming jargon). After all, the ultimate accomplishment for programmers is creating software that never breaks; but programming that doesn't account for edgecases is likely to do exactly that under those extreme circumstances. So programmers are trained to treat every case equally, whether it represents one user or one thousand.

This approach doesn't work for interface design. An interface that gives a small number of extreme cases the same weight as the needs of the vast majority of users ends up ill-equipped to make either audience happy. A well-designed interface recognizes the courses of action users are most likely to take and makes those interface elements easiest to access and use.

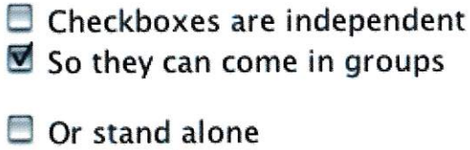


This doesn't mean that the solution to every interface problem is to make the button users are most likely to push the biggest one on the page. Interface designs can employ a variety of tricks to ease users along the way to their goals. One simple trick is to think carefully about the default options selected when the interface is first presented to the user. If your understanding of your users' tasks and goals leads you to think most of them would prefer detailed search results over brief ones, leaving the "Show Me More Detail" checkbox checked by default means more people will automatically be happy with what they get, regardless of whether they took the time to read the label on the checkbox and make a decision for themselves. (Even better is a system that automatically remembers the options a user selected the last time they stopped by, but this sometimes requires more technical acrobatics than would appear necessary on the surface, and as a result is impractical for some development teams to implement successfully.)

The two primary technologies through which interfaces are delivered on the Web, HTML and Flash, have inherent technical constraints that limit the interface options available to us. This is actually both bad and good. It's bad because it limits our opportunities for innovation—some interface approaches that might be possible with traditional desktop software are simply impossible to realize on the Web. But this situation is also good, because users who learn how to work with a fairly small set of standard controls can apply that knowledge across a wide range of sites.

Although HTML was originally intended for simple hypertext information, people quickly saw its potential to provide more interactivity. Not long after its release into the world, HTML acquired a handful of standard interface elements:

**Checkboxes** allow users to select options independently of one another.



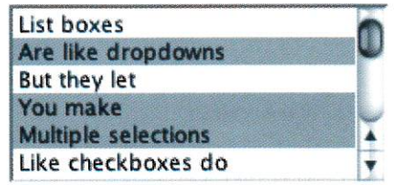

**Radio buttons** allow users to select one option from among a set of mutually exclusive options.


**Text fields** allow users to—wait for it—enter text.




**Dropdown lists** provide the same functionality as radio buttons, but they do so in a more compact space, allowing many more options to be presented efficiently.


**List boxes** provide the same functionality as checkboxes, but they do so in a more compact space (because list boxes scroll). As with dropdowns, this enables the list box to easily support a large number of options.




**Action buttons** can do lots of different things. Typically, they tell the system to take all the other information the user has provided via other interface elements and do something—take action—with it.

Flash provides the same set of basic elements, but because of its origins as an animation tool, it can offer a greater degree of flexibility in how the interface can respond to the user. Consequently, Flash interfaces involve a lot more choices to be made during the design process, and they tend to be harder to get right.

Juggling all the different interface elements and choosing from among them inevitably involves tradeoffs. True, that dropdown will save you some space on the page relative to a set of radio buttons, but it will also hide the available choices from the user. Having people type in the categories they want to search might put less load on the database, but it puts more load on the user; if there are only six to choose from anyway, maybe some checkboxes would be better.

Interface design for the Web is all about selecting the right interface elements for the task the user is trying to accomplish and arranging them on the page in a way that will be readily understood and easily used. Tasks on Web sites will typically stretch across several pages, each containing a different set of interface elements for the user to contend with. Which functions end up on which pages is a matter of interaction design down in the structure plane; how those functions are realized on the page is the realm of interface design.


Dropdown lists can hinder users by hiding important options from view (left). Radio buttons easily display all the available options (right), but they take more space in the interface.



Information design plays a role in interface design problems when the interface must not only gather information from the user, but communicate information to the user as well. Error messages are a classic information design problem in creating successful interfaces; providing instructional information is another one, if only because the biggest challenge is getting users to actually read the instructions. Any time the system has to give the users some information for them to use the interface successfully—whether it's because they made a mistake or because they're just getting started—that's an information design problem.

## Navigation Design

Navigation design seems like a simple business: Put links on every page that allow users to get around on the site. If you scratch the surface, however, the complexities of navigation design become apparent. The navigation design of any site must accomplish three simultaneous goals:

- First, it must provide users with a means for getting from one point to another on the site. Because it's usually impractical (and even when practical, it's generally not a good idea) to link to every page from every other page, navigation elements have to be selected to facilitate real user behavior—and by the way, that means those links have to actually work, too.

- Second, the navigation design must communicate the relationship between the elements it contains. It's not enough to merely provide a list of links. What do those links have to do with each other? Are some more important than others? What are the relevant differences between them? This communication is necessary for users to understand what choices are available to them.
- Third, the navigation design must communicate the relationship between its contents and the page the user is currently viewing. What does any of this stuff have to do with what I'm looking at right now? Communicating this helps users understand which of the available choices might best support the task or goal they are pursuing.

In a physical space, people can rely on some degree of an innate sense of direction to orient themselves. (Of course, some people just seem to be perpetually lost.) But the mechanisms in our brain that help us find our way around in the physical world ("Let's see, I think the entrance where I came in should be behind me and to the left") are utterly useless in helping us find our way around in an information space.

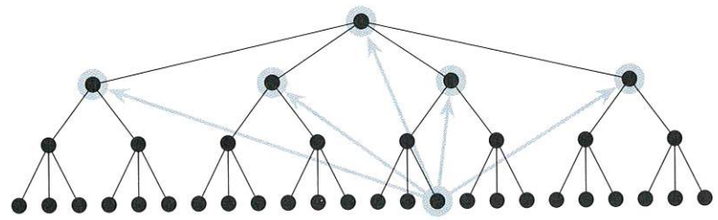
That's why it's of vital importance that every page of a Web site communicate clearly to users where they are on the site and where they can go. The question of to what extent users orient themselves in information spaces is currently a matter of some debate: Some people strongly favor the notion that users make little maps in their heads when they visit Web sites, just as they do in hardware stores and libraries; others claim users rely almost entirely on the navigational and wayfinding cues in front of them, as if each step they took through the site faded from their memory shortly after they took it.



We still don't know just how (or how much) people keep the structure of Web sites in their heads. Until we figure it out, the best approach is to assume that users carry no knowledge with them from page to page. (After all, if a public search engine such as Google indexes your site, any page could be an entry point to your site anyway.)

Most sites actually provide multiple **navigation systems**, each fulfilling a particular role in enabling the user to navigate the site successfully in a variety of circumstances. Several common types of navigation systems have sprung up in practice.

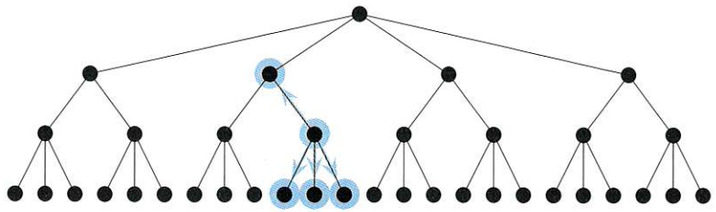
**Global navigation** provides access to the broad sweep of the entire site. The use of the term "global" here doesn't necessarily imply that this navigation appears on every page in the site—although that's not a bad idea. (We use the term "persistent" to refer to navigation elements that appear throughout a site; again, keep in mind that persistent elements are not necessarily global.) Instead, global navigation brings together the key set of access points that users might need to get from one end of the site to the other. Anywhere you might want to go, you can get there (eventually) from the global navigation.



Global navigation.

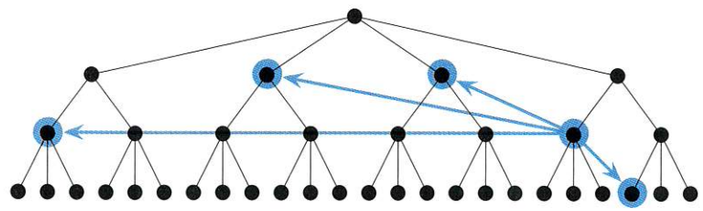
**Local navigation** provides users with access to what's "nearby" in the architecture. In a strictly hierarchical architecture, local navigation might provide access to a page's parent, siblings, and children. If your architecture is constructed to reflect the ways users think about the site's content, local navigation will typically get more use than other navigation systems.

Local navigation.



**Supplementary navigation** provides shortcuts to related content that might not be readily accessible through the global or local navigation. This type of navigation scheme offers some of the benefits of faceted classification (allowing users to shift the focus of their exploration of the content without starting over at the beginning), while still permitting the site to maintain a primarily hierarchical architecture.

Supplementary navigation.

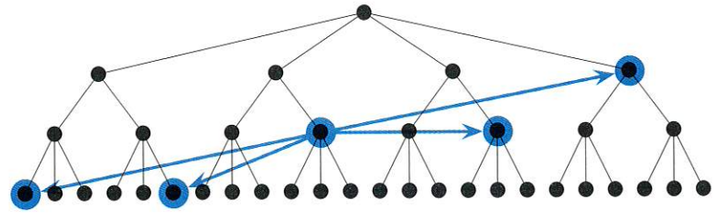




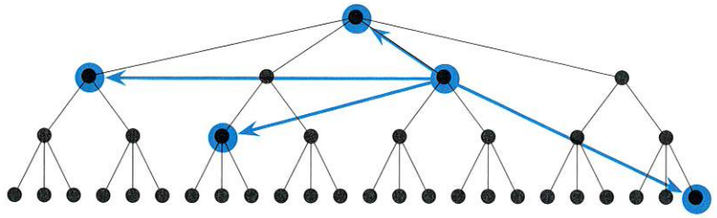
**Contextual navigation** (sometimes called "inline navigation") is embedded in the content of the page itself. This type of navigation—for example, a hyperlink within the text of a page—often goes under- (or mis-) utilized. Often, the moment when users decide they need a different piece of information is when they're reading the text. Instead of forcing your users to scan the page for the right navigation element—or worse, sending them scurrying to the search engine—why not put the relevant link right there?

Reaching all the way back to the strategy plane, the better you understand your users and their needs, the more effectively you can deploy contextual navigation. If it doesn't clearly support your users' tasks and goals—if your text is crammed full of so many hyperlinks that users can't pick out what's relevant to their needs—contextual navigation will (rightly) be seen as clutter.

Contextual navigation.



Courtesy navigation.



**Courtesy navigation** provides access to items that users don't need on a regular basis, but that are commonly provided as a convenience. In the physical world, a retail store will usually post its hours of operation at its entrance. For most customers, most of the time, this information isn't all that helpful: Anybody can tell pretty quickly whether or not the shop is open for business. But knowing that the information is readily available helps them when they do need it. Links to contact information, feedback forms, and policy statements are commonly found in courtesy navigation.

Some navigational devices aren't embedded within the structure of your pages, but stand on their own, independent of the content or functionality of your site. These are **remote navigation** tools that users turn to most often when they get frustrated with the other navigational systems you've provided, or when they've taken one look at your navigational systems and quickly come to the conclusion that they're better off not even attempting to figure them out.



A **site map** is a common remote navigation tool that gives users a concise, one-page snapshot of the overall site architecture. The site map is usually presented as a hierarchical "outline" of the site, providing links to all the top-level sections with links to major second-level sections indented beneath them. Site maps don't usually show more than two levels of the hierarchy—beyond that is more detail than users typically need (and if it isn't, there's probably something wrong with your high-level architecture).

An **index** is an alphabetical list of topics with links to relevant pages, much like the index in the back of a book. This type of tool is most effective for sites that have a great deal of content covering a diverse range of subjects. In most other cases, a site map and a well-planned architecture should be sufficient. Indexes are sometimes developed for individual sections of a site, rather than attempting to cover the entire sweep of the site's content; this approach can be useful when you have sections intended to serve different audiences with divergent information needs.

## Information Design

Information design can sometimes be difficult to put your finger on. It often serves as the glue that holds the other components of the design together. In all cases, information design comes down to making decisions about *how to present information* so that people can use it or understand it more easily.

Sometimes information design is visual. Is a pie chart the best way to present that data, or would a bar chart work better for our users? Does the "binoculars" icon adequately convey the concept of searching the site, or would a magnifying glass be better understood?

Sometimes information design involves grouping or arranging pieces of information. We often take this aspect of design for granted because we are used to seeing common information grouped in certain ways. For example, look at this list of items:

- State
- Job title
- Telephone number
- Street address
- Name
- Zip code
- Organization
- City
- E-mail address

It seems a little confusing, because usually it looks more like this:

- Name
- Job title
- Organization
- Street address
- City
- State
- Zip code
- Telephone number
- E-mail address



Even this arrangement could be clarified further:

- Personal information
  - Name
  - Job title
  - Organization
- Address information
  - Street address
  - City
  - State
  - Zip code
- Other contact information
  - Telephone number
  - E-mail address

This example seems pretty straightforward, but a slightly different list of items could prove more challenging:

- Power limit
- Rotor size
- Tank capacity
- Transmission type
- Median angular velocity
- Chassis style
- Maximum output

The key, of course, is to group and arrange the information elements in ways that reflect how your users think and support their tasks and goals. The conceptual relationships between these elements really amount to micro-level information architecture; information design comes into play when we have to *communicate* that structure on the page.

### Wayfinding

One important function that information design and navigation design work together to perform is supporting **wayfinding**. The idea of wayfinding comes from the design of public spaces in the physical world. Parks, stores, roads, airports, and parking lots all benefit from the incorporation of wayfinding devices. Parking garages, for example, will sometimes use color coding to give people cues to help them remember where they left their cars. In airports, signs, maps, and other indicators help people find their way around.

On Web sites, wayfinding typically involves both navigation design and information design. The navigation systems employed by a site not only have to provide access to the different areas of the site, they also have to communicate those choices successfully. Good wayfinding enables users to quickly get a mental picture of where they are, where they can go, and which choices will get them closer to their objectives.



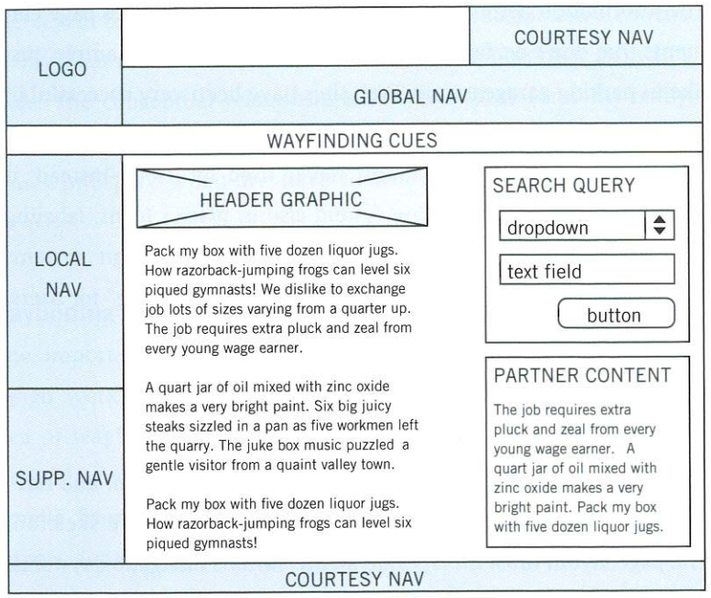
The information design component of wayfinding involves page elements that don't perform a navigational function. For example, just like in parking garages, some Web sites have been very successful in using color-coding to indicate which section a user is looking at. (However, color-coding is almost never used by itself—instead, it reinforces another wayfinding system also in place.) Icons, labeling systems, and typography are other information design systems sometimes used to help reinforce a sense of "you are here" for users.

## Wireframes

Page layout is where information design, interface design, and navigation design come together to form a unified, cohesive skeleton. The page layout must incorporate all the various navigation systems, each designed to convey a different view of the architecture; all the interface elements required by any functionality on the page; and the information design that supports both of these, as well as the information design of the page content itself.

It's a lot to balance all at once. That's why page layout is covered in detail in a document called a page schematic or **wireframe**. The wireframe is a bare-bones depiction (as the name suggests) of all the components of a page and how they fit together.

Wireframes capture all the skeleton decisions in a single document that serves as a reference for visual design work and site implementation. Wireframes can contain varying levels of detail—this one is pretty light.




This simple line drawing is usually heavily annotated, referring the reader to architecture diagrams or other interaction design documentation, content requirements or functional specifications, or other types of detailed documentation as needed. For example, if a wireframe refers to specific existing content elements, it might provide pointers to where they can be found. In addition, the wireframe will often contain supplementary notes on intended site behavior that might not be obvious from just looking at the wireframe and the architecture diagram.



In many ways, the architecture diagram we saw back in the structure plane is the grand vision for the project; here in the skeleton plane, the wireframe is the detailed document that shows just how that vision will be fulfilled. Wireframes will sometimes be supplemented by comprehensive navigational specifications, describing in more detail the precise composition of each of the various navigational components.

For smaller or less complex sites, a single wireframe is sufficient to serve as a template for all the pages that will be built. For many projects, however, multiple wireframes are needed to convey the complexity of the intended result. You probably won't need a wireframe for each page of the site, however. Just as the architectural process allowed us to classify content elements into broad classes, a relatively small number of standard page types will emerge from wireframe development.

Wireframes are a necessary first step in the process of formally establishing the visual design for the site, but almost everyone involved in the development process will use them at some point. People responsible for strategy, scope, and structure can refer to the wireframe to confirm that the final product will meet their expectations. People responsible for actually building the site can refer to the wireframe to answer questions about how the site should function.

As the field of user experience development has grown and evolved, responsibility for the wireframe has at times been the subject of a sort of turf war inside organizations. Some Web development teams are highly compartmentalized to the point that they have distinct roles (and sometimes entire departments!) for "information architects" and "designers."

The wireframe, being the place where information architecture and visual design come together, then becomes a subject of debate and dispute. Information architects complain that designers who create wireframes obscure their architectures behind navigation systems that don't reflect the principles underlying the architectures. Visual designers complain that wireframes produced by information architects reduce their role to that of paint-by-numbers artist, squandering the experience and expertise in visual communication they bring to information design problems.

When you have separate information architects and designers, the only way to produce successful wireframes is through collaboration. The process of having to work out the details of the wireframe together enables each side to see issues from the other's point of view, and it can help uncover problems early in the process (instead of later, when the site is being built and everyone is wondering why it isn't working as planned).

All of this makes wireframes sound like a whole lot of work. They don't have to be. Documentation is never an end in itself; it is only a means to an end. Creating documentation for its own sake is not merely a waste of time—it can be counter-productive and demoralizing. Producing the right level of documentation for your needs—and not fooling yourself that you can get by with less—turns documentation from a problem into an advantage.



Some of the most successful wireframes I've ever worked with have been nothing more than pencil sketches with Post-It Notes attached. For a small team in which the designer and the programmer sit right next to each other, that level of documentation is perfectly sufficient. But when programming is the responsibility of an entire team and not just one person—and that team is halfway around the world—something a bit more formal is probably called for.

The value of wireframes is the way they integrate all three elements of the structure plane: interface design, through the arrangement and selection of interface elements; navigation design, through the identification and definition of core navigational systems; and information design, through the placement and prioritization of informational components. By bringing all three together into a single document, the wireframe can define a skeleton that builds on the underlying conceptual structure while pointing the way forward toward visual design.

## Further Reading

Fleming, Jennifer. *Web Navigation: Designing the User Experience*. O'Reilly, 1998.

Spolsky, Joel. *User Interface Design for Programmers*. Apress, 2001.

Tufte, Edward. *Envisioning Information*. Graphics Press, 1990.

Veen, Jeffrey. *The Art & Science of Web Design*. New Riders, 2000.

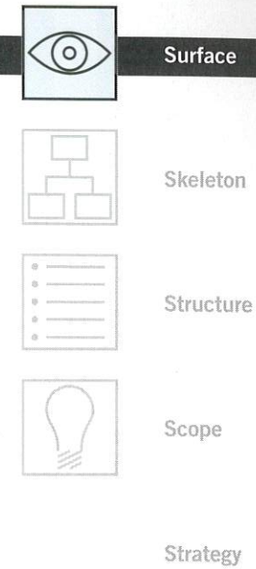
Web resources: **www.jjg.net/elements/resources/**.

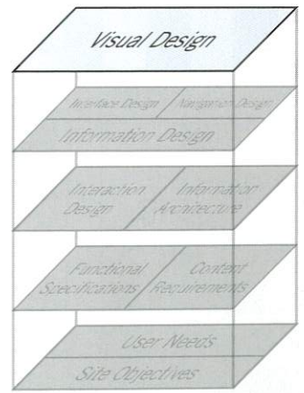
chapter 7

# The Surface Plane

## Visual Design

At the top of the five-plane model, we turn our attention to those aspects of the site our users will notice first: the visual design. Here, content, functionality, and aesthetics come together to produce a finished design that fulfills all the goals of the other four planes.

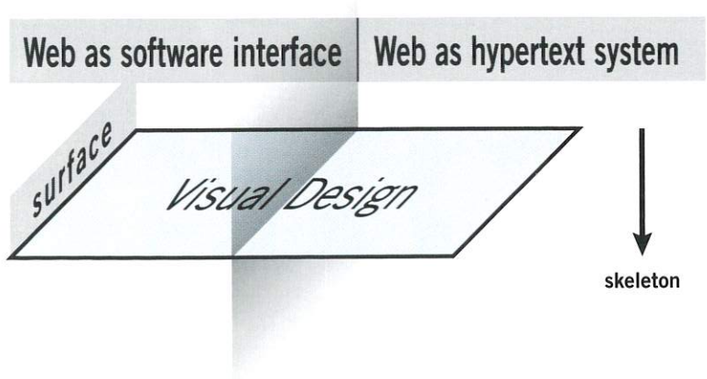


## Defining the Surface

On the skeleton plane, we were dealing primarily with arrangement. Interface design concerns the arrangement of elements to enable interaction, navigation design concerns the arrangement of elements to enable movement through the site, and information design concerns the arrangement of elements to communicate information to the user.

Moving up to the surface plane, we are now dealing with the visual presentation of the logical arrangements that make up the skeleton of the site. For example, through attention to information design, we determine how we should group and arrange the information elements of the page; through attention to **visual design**, we determine how that arrangement should be presented visually.






Initially, you might think visual design is a simple matter of aesthetics. Everybody has different taste, and everybody has a different idea of what constitutes a visually appealing design, so every argument over design decisions just comes down to personal preference, right? Well, everybody does have a different sense of aesthetics, but that doesn't mean design decisions have to be based on what looks "cool" to everyone involved.

Instead of evaluating visual design ideas solely in terms of what seems aesthetically pleasing, you should focus your attention on how well they work. How effectively does the design support the objectives defined by each of the lower planes? Does the look of the site undermine the structure, making distinctions between sections of the architecture unclear or ambiguous? Or does the visual design reinforce the structure, clarifying the options available to users?

Communicating a brand identity, for example, is a common strategic objective for a Web site. Brand identity comes across in many ways—in the language you use or in the interaction design of your site's functionality—but one of the main tools used to communicate brand identity is visual design. If the identity you want to convey is technical and authoritative, using comic-book fonts and bright pastel colors probably isn't the right choice. It's not just a matter of aesthetics, it's a matter of strategy.

## Follow the Eye

One simple way to to evaluate the visual design of a page is to ask: Where does the eye go first? What element of the design initially draws the user's attention? Are they drawn to something important to the site's strategic objectives? Or is the first object of their attention a distraction from their goals (or yours)?

Researchers sometimes use sophisticated **eyetracking** equipment to determine exactly what test subjects are looking at and how their eyes move about the page. If you're just fine-tuning the visual design of a page, however, you can usually get away with simply asking people—even just asking yourself. Sometimes this approach won't provide the most accurate results, and it will never capture all the nuances that eyetracking equipment can provide. But most of the time, simply asking questions is perfectly suitable. Another way to find the dominant design element is to squint at the page or blur your eyes until you can't make out the details—or walk to the other end of the room and look at it from there.

Then try to determine where the eye goes. If you're serving as your own test subject, pay attention to the unconscious movements of your eyes around the page. Don't think too much about what you're looking at; just let your eyes take in the page naturally. If someone else is your test subject, ask them to call out the elements of the page in the order that their attention is drawn to them.

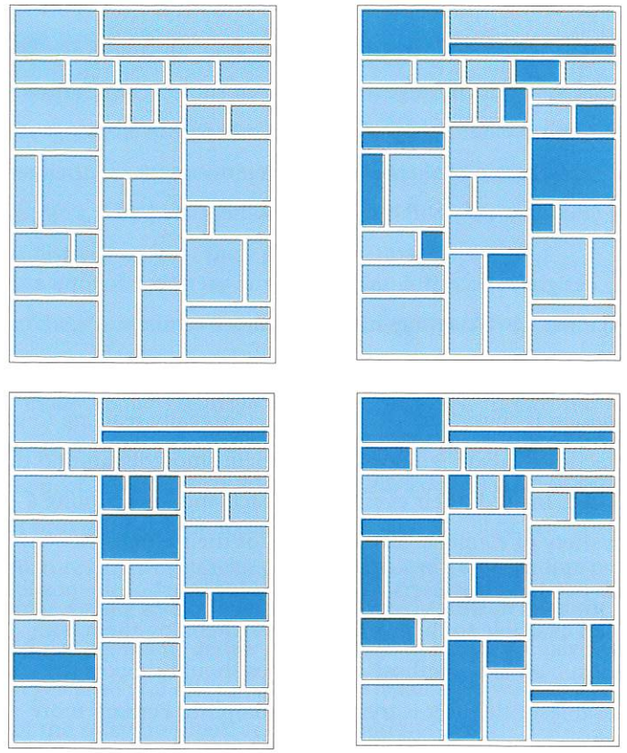


Generally, you'll find that people follow fairly consistent patterns in how their eyes move across the page—after all, these are largely unconscious, instinctive eye movements. If subjects report their eyes following a very different pattern from other people, they probably aren't really aware of their natural eye movements, or they're just telling you what they think you want to hear (or both).

If your design is successful, the pattern the user's eye follows around the page will have two important qualities:

- First, it follows a smooth flow. When people comment that a design is "busy" or "cluttered," they're really reacting to the fact that the design doesn't lead them smoothly around the page. Instead, their eyes bounce back and forth among a variety of elements all clamoring for their attention.
- Second, it gives users a sort of "guided tour" of the possibilities available to them without overwhelming them with detail. As always, those possibilities should support the goals and tasks the user is trying to accomplish. Perhaps more importantly, those possibilities shouldn't distract from information or functions that users will need to fulfill those goals.

The movement of the user's eyes around the page doesn't happen by accident. It's the result of a complex set of deeply ingrained instinctive responses to visual stimuli that all humans share. Fortunately for us designers, these responses are not completely outside our control either—over the centuries, we've developed a variety of effective visual techniques for attracting and directing attention.



In a visually neutral layout (top left), nothing stands out. Contrast can be used to guide the user's eye around the page (top right) or draw their attention to a few key elements (bottom left). Overuse of contrast leads to a cluttered look (bottom right).

## Contrast and Uniformity

In visual design, the primary tool we use to draw the user's attention is **contrast**. A design without contrast is seen as a gray, featureless mass, causing the user's eyes to drift around without settling on anything in particular. Contrast is vital to drawing the user's attention to essential aspects of the interface, contrast helps the user understand the relationships between the navigational elements on the page, and contrast is the primary means of communicating conceptual groups in information design.



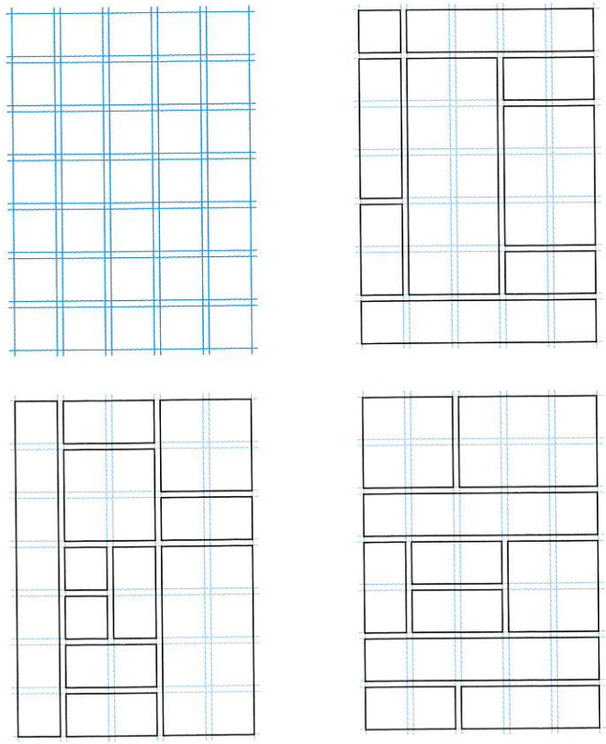
When elements in a design are different, users pay attention. They can't help it. You can use this instinctive behavior to your advantage by making the pieces users really need to see stand out from the rest of the elements on the page. Error messages in Web interfaces often suffer from blending in with the rest of the page; contrasting them by putting the text in a different color (like, say, red) or highlighting them with a bold graphic can make all the difference.

For this strategy to work, however, the difference has to be significant enough for the user to clearly tell that the design choice is intended to communicate something. When the design treatment of two elements is similar but not quite the same, users get confused. "Why are those different? Are they supposed to be the same? Maybe it was just a mistake. Or am I supposed to notice something here?" Instead, we want both to grab users' attention and to assure them that it was intentional.

Maintaining **uniformity** in your design is an important part of ensuring that your design communicates effectively without confusing or overwhelming your users. Uniformity comes into play in many different aspects of visual design.

Keeping the sizes of elements uniform can make it easier to recombine them into new designs as you need them. For example, if all the graphic buttons you use for navigation are the same height, they can be mixed and matched as needed without creating a cluttered layout or requiring that new graphics be produced.

**Grid-based layout** is one technique from print design that carries over effectively to the Web. This approach ensures uniformity of design through a "master layout" that is used as a template for creating layout variations. Not every layout will use every part of the grid—in fact, most layouts will probably only use a few—but every element's placement on the grid should be uniform and consistent. However, because Web browsers don't afford complete control of the size of text elements, applying grids to the Web isn't always as simple as it is in print design.



Creating a grid to guide layout work enables uniformity without sacrificing consistency.



It's easy to fall into the trap of adhering to a grid system—or any standard intended to ensure uniformity—even when it clearly isn't working anymore. The anarchy of working without design standards is bad, but the straitjacket of trying to work within design standards that are inadequate for your needs can be worse. Maybe the site has taken on new functionality that no one had imagined at the time when the grid was developed; maybe the grid just never worked all that well in the first place. Whatever the reason, it's important to be able to recognize when it's time to revisit the foundations of your design system.

## Internal and External Consistency

Because of the way Web sites often have been produced—piecemeal, ad-hoc, and isolated from other design work going on in the organization—they have been plagued with problems of consistency in visual design. These problems take two forms:

- First, there are problems of internal consistency, in which different parts of the site reflect different design approaches.
- Then there are problems of external consistency, in which the site doesn't reflect the same design approach used in other products from the same organization.

Good solutions to problems of internal consistency are rooted in an understanding of the skeleton of the site. Identify recurring design elements that appear in different contexts throughout the various interface, navigation, and information design problems on the site. By removing each design element from those different contexts before we design it, we can more clearly see the small-scale problem we're trying to solve, instead of getting distracted by the larger-scale problems imposed by context. Rather than designing the same element over and over again, we can examine it in isolation, design it once, and use that design throughout the site.

Obviously, for such an approach to work, we will still have to check our work against the different contexts in which that element appears. Maybe a big, round, red "STOP" button will work fine for the checkout page, but it might not work so well on the crowded product customization page. The best approach is to isolate each element, design it, try the design in various contexts, and then rework the design as needed.

Even though many of the design elements will be created in isolation from each other, they should still work together. A successful design is not merely a collection of small, well-designed objects; rather, the objects should form a *system* that operates as a cohesive, consistent whole.



Historically, problems of internal consistency often came about because many Web sites grew out of corporate IT departments, which generally had little understanding of (and even less access to) company visual design standards. In the early days, this divide was only reinforced by the profound technical limitations that the new medium placed on design. Marketing departments, accustomed to the nearly infinite control afforded by designing for print and broadcast media, essentially disowned Web design as being too primitive to be worth their resources. IT departments, left to their own devices, learned to accommodate the limits of the new medium and gradually developed their own design standards.

These days, however, most marketing departments have more direct control over the look of their companies' Web sites. Also, advances in technology and more sophisticated design techniques have enabled Web sites to look more like their print and broadcast counterparts than ever. As a result, sites sporting a radically different look from the rest of the material produced by their owners are growing increasingly rare.

You still can't expect to be able to do everything with Web design that you can do with design in other media. Nevertheless, you can gain a lot of value from making sure your online and offline styles align wherever possible. This doesn't mean that the styles must be precisely the same—instead, they should be constructed so as to produce the same effect.

Enforcing design consistency across media presents your audience—customers, prospects, shareholders, employees, or casual observers—with a uniform impression of your brand identity. This consistency of brand identity plays out at every level of the design of your site, from the navigation elements appearing across every page to the humble button that appears only once.

Presenting a style on your Web site that's inconsistent with your style in other media doesn't just affect the audience's impression of the site; it affects their impression of your company as a whole. People respond positively to companies with clearly defined identities. Inconsistent visual styles undermine the clarity of your corporate image and leave the audience with the impression that this is a company that hasn't quite figured out who it is.

## Color Palettes and Typography

Color can be one of the most effective ways to communicate a brand identity. Some brands are so closely associated with colors that it's difficult to think of the company without the color automatically coming to mind—consider Coca-Cola, UPS, or Kodak. These companies have employed specific colors (red, brown, yellow) consistently over the years to create a stronger sense of their identities in the public's mind.

That doesn't mean they use these colors to the exclusion of all others. The core brand colors are usually part of a broader **color palette** used in all of a company's materials. The colors in a company's standard palette are selected specifically for how well they work together, complementing each other without competing.

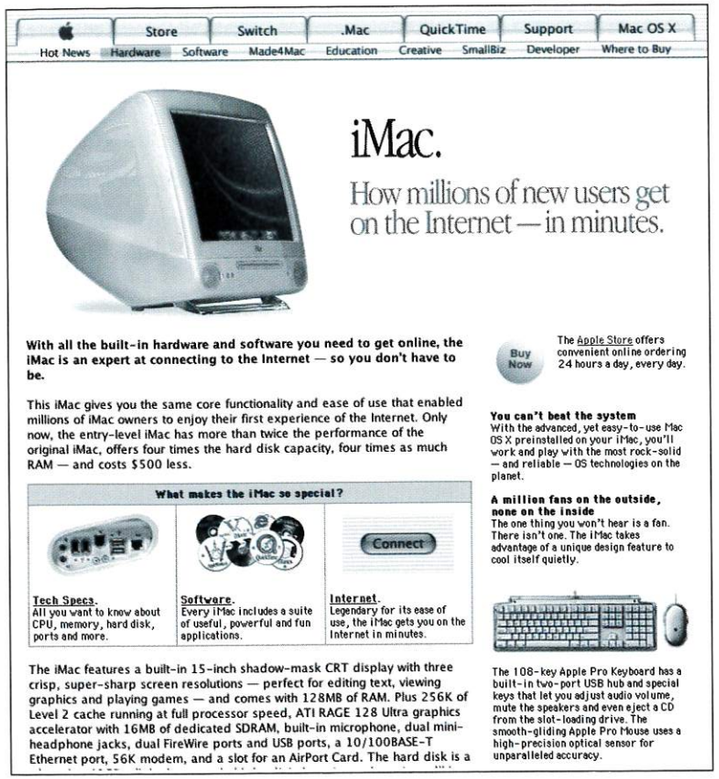


A color palette should incorporate colors that lend themselves to a wide range of uses. In most cases, brighter or bolder colors can be used for the foreground of your design—elements to which you want to draw attention. More muted colors are better used for background elements that don't need to jump off the page. Having a range of colors to choose from provides us with a toolkit for making effective design choices.

Just as contrast and uniformity are important to other areas of visual design, they play a vital role in the creation of color palettes as well. When used in the same context, colors that are very close to one another, but not quite the same, undermine the effectiveness of your color palette. This doesn't mean you only get one shade of red, one shade of blue, and so forth. It means that if you want to use different shades of red, make sure they're different enough that users can tell them apart, and make sure you use each in consistent ways.

For some companies, **typography**—the use of fonts or typefaces to create a particular visual style—is so important to their brand identities that they have commissioned special typefaces to be produced specifically for their use. Organizations ranging from Apple Computer to Volkswagen to the London Underground have used custom typography to create a stronger sense of identity in their communications. But even if you choose not to take this extraordinary step, type can still serve as an effective part of communicating your identity through visual design.

Like many organizations, Apple Computer uses consistent **typography**—both on its Web site and in other media—to convey a consistent brand image.






Because of the limited resolution of computer screens, some typefaces that would be perfectly acceptable on paper can be very difficult to read on a Web site. For this reason, typefaces designed expressly to be highly readable on screen (such as Microsoft's Georgia and Verdana fonts) have become popular alternatives to less legible "default" fonts such as Arial or Times New Roman.

For larger text elements or short labels like those seen on navigational elements, typefaces with a little more personality are perfectly appropriate. But one of our objectives is not to overwhelm our users with visual clutter, and using an unnecessarily wide variety of fonts—or even using a small number of fonts in inconsistent ways—can contribute to that sense of clutter. In most cases, you won't need more than a handful of fonts to meet all your communication needs.

The principles of using type effectively are really the same as those for other aspects of visual design: Don't use styles that are very similar but not exactly the same. Use different styles only to indicate differences in the information you're trying to communicate. Provide enough contrast between styles that you can draw the user's attention as needed, but don't overload the design with a wide range of diverse styles.

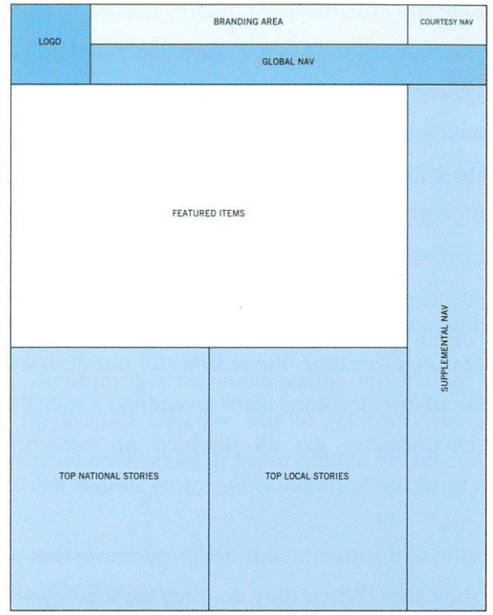
## Design Comps and Style Guides

The most direct analogue to the wireframe for the realm of visual design is the visual mock-up or **design comp**. The "comp" is short for composite, because that's exactly what it is: a visualization of the finished product built up from the components that have been chosen. The comp shows how all the pieces work together to form a cohesive whole; or, if they don't, it shows where the breakdown is happening and demonstrates constraints that any solution will have to account for.

You should be able to see a simple one-to-one correlation between components of the wireframe and components of the design comp. The comp might not faithfully reproduce the layout of the wireframe—in most cases it probably won't. The wireframe doesn't account for visual design concerns, focusing instead on documenting the skeleton. Building the wireframe before we tackle the design comp allows us to look at skeleton issues in isolation first, then see how surface issues come into play. Nevertheless, the conceptual aspects of the wireframe, particularly regarding information design issues, should be plainly evident in the design comps, even though they may not follow the precise arrangement presented in the wireframe.

The definitive documentation of all the design decisions we have made is the **style guide**. This compendium defines every aspect of the visual design, from the largest scale to the smallest. Global standards affecting every page of the site—such as design grids, color palettes, typography standards, or logo treatment guidelines—are usually the first things to go into a style guide.







The visual design doesn't have to match the wireframe precisely—it only has to account for the relative importance and grouping of elements presented in the wireframe.

The style guide will also include standards specific to a particular section or function of a site. In some cases, the standards documented in the style guide will go all the way down to the level of individual interface and navigation elements. The overall goal of the style guide is to provide enough detail to help people make smart decisions in the future (because most of the thinking has already been done for them).

All of this documentation is, of course, a lot of work, but it happens for a good reason: Over time, the reasons for our decisions fade from memory. The ad-hoc decisions made to address a specific problem in a specific circumstance get all jumbled up with the decisions intended to form the foundation for future design work.

Another reason to document your design system is that people eventually quit their jobs. When they do, they walk away with a wealth of knowledge about how a site gets designed and built on a day-to-day basis. Without a style guide that remains up-to-date with the latest standards and practices, that knowledge is lost. Over time, as people change positions, the whole organization gradually suffers a sort of amnesia, as the ways things were done and the reasons for those decisions drift away to other parts of the company or back out into the workforce.



Creating a style guide is also helpful in imposing design consistency across a decentralized organization. If your Web operations consist of a diverse range of independent projects being initiated and worked on by people in offices scattered all over the world, your site is likely to look like a random mish-mash of styles and standards. Getting all those people to go along with a unified set of standards can be a lot of work, which is why responsibility for enforcing design style guides often resides higher up in the organization than you might expect. A single style guide, designed to take the needs of all these different projects into account, is a big undertaking, but it's not impossible, and it's the single most effective way to get your Web site looking like a coherent whole instead of just a jumble of tacked-on pieces.

## Further Reading

Mullet, Kevin and Darrell Sano. *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall, 1994.

Williams, Robin. *The Non-Designer's Design Book*. Peachpit, 1994.

Web resources: **www.jjg.net/elements/resources/**

chapter 8

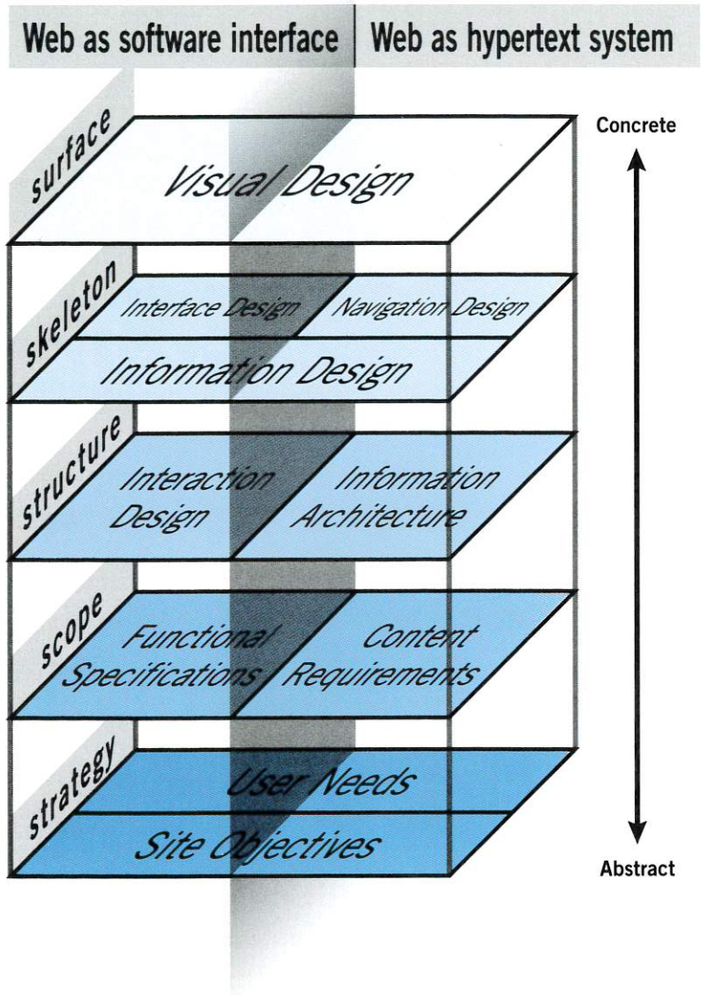
# The Elements Applied



The elements of user experience remain consistent no matter how big your site is. But putting the ideas behind the elements into practice can sometimes seem like a challenge all by itself. It's not just a question of time and resources—it's often a question of mindset.

Looking back over the five planes—strategy, scope, structure, skeleton, and surface—it all sounds like a whole lot of work. Surely such careful attention to all these details must take months of development time and a small army of highly trained specialists, right?

Not necessarily. Certainly, for some projects and some organizations, employing a team of dedicated specialists is the most effective way to parcel out responsibility for a site that's simply too complex to handle any other way. Also, because specialists can focus exclusively on a subset of the complete user experience, they often bring a deeper understanding of those issues to bear in their work. Much of the time, however, small teams with limited resources can achieve similar results. Sometimes a group of just a few people can actually produce better results than a large team.






Creating the user experience is really little more than a very large collection of very small problems to be solved. The difference between a successful approach and one doomed to failure really comes down to two basic ideas:

- **Understand what problem you're trying to solve.** So you've worked out that the big purple button on the home page is a problem. Is it the bigness and the purpleness of the button that needs to change (surface)? Or is it that the button is in the wrong place on the page (skeleton) or that the function the button represents doesn't work the way users expect it to (structure)?
- **Understand the consequences of your solution to the problem.** Remember that there's a potential "ripple effect" up and down through the elements from every decision you make. The navigation design that works so well in one part of your site might not quite meet the needs of another section of the architecture. The interaction design for the product selection wizard might well be an innovative approach, but will it meet the needs of your technophobic users?

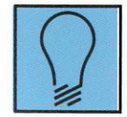
This might seem to be a painfully obvious approach to the task of creating the user experience of your site, but you'd be surprised at just how many of the tiny decisions that make up the user experience development process aren't made consciously at all. Most of the time, the choices made about the user experience fall into one of these scenarios:

- **Design by default.** This happens when the structure of the user experience follows the structure of the underlying technology or of your organization. Keeping the customer's order history and billing information in separate databases might work better for your existing technical system, but that doesn't mean keeping them separate in the user experience is also a good idea. Similarly, content that comes from different departments in the company might serve the user better if it were brought together, not kept separate.
- **Design by mimicry.** This happens when the user experience falls back on familiar conventions from other sites, publications, or software applications, regardless of how appropriate those conventions might be to your users (or even to the Web itself). The widespread, haphazard adoption of tabs as a global navigation device in the late 1990s is one example of this phenomenon.
- **Design by fiat.** This happens when personal preferences drive user experience decisions instead of user needs or site objectives. If orange dominates your color palette because one of the senior vice presidents is fond of it, or if all your navigational elements are drop-down menus because that's what your head engineer likes, you've lost sight of the strategic goals that should be driving the choices you make.

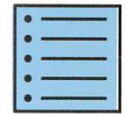


## An Example: Search Engine Implementation

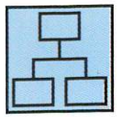
Search engines are probably the most common feature anywhere on the Web—everyone uses them at one time or another, and sites that don't offer search engines are increasingly rare. But even though they seem to be everywhere, these apparently simple tools require a complex set of decisions spanning every plane of the elements of user experience if they are to be implemented successfully.



The nearly ubiquitous inclusion of search engines on Web sites reflects the growing understanding that the capability to retrieve content matching specific criteria is a practically universal user need, regardless of the specific audience or content in question. Addressing this user need is a key **strategy** decision.

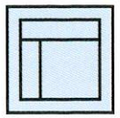


The site's content requirements and functional specifications dictate the **scope** of what the search engine can offer users. If having metadata about your content is a requirement for the site, the search engine can provide users with enhanced capabilities that make use of that data, such as enabling them to search only for articles by a particular author or those published within a certain time period. If some of your metadata is not intended to be used by the search engine, the functional specifications can detail exactly what types of searches users can perform.



The search engine really starts to take shape when **structure** issues—interaction design and information architecture—are taken into consideration. The interaction design of the search function determines how users will actually work with the search engine. Perhaps the search functionality is so complex that it will require a lengthy, structured process before the user even gets to see results; on the other hand, maybe all you need is a keyword entry field on every page.

Also, if the search engine is aware of the information architecture of the site as a whole, it can offer more capabilities to the user, such as the option to restrict a search to one particular area of the site or to automatically sort search results according to their placement in the architecture. In addition, the search results will have an information architecture of their own. Do all results appear on a single page, or are they broken out across multiple pages? If the latter, how does the user get from one page to another? Can he jump around arbitrarily, or is he limited to a strictly sequential navigation path?



These structural decisions take concrete form in the **skeleton** of the search engine. The interaction design takes shape in the arrangement of buttons, fields, and other interface elements that enable the user to submit a query to the system. The information architecture is reflected in the design of the navigational elements that allow the user to move through search results. Throughout, information design shows users how to construct queries and enables them to scan through search results to find what they need.



On the **surface**, all these elements are bound together into a cohesive whole by the search engine's visual design, which gives interface and information elements alike a consistent look and draws attention to the areas of the page in which users are likely to be most interested (or that they would otherwise miss).

## Asking the Right Questions

Facing the tangle of small problems to be solved in creating the user experience can sometimes be disheartening. Occasionally a solution to one problem will force you to rethink other problems you thought you had already solved. Many times, you will have to make compromises and evaluate tradeoffs between different approaches. When you're in the middle of having to make these kinds of decisions, it's easy to become frustrated and question whether you're taking the right approach. The simple fact is this: The right approach is one in which no aspect of the user's experience is left to chance. Make every decision consciously and deliberately and ground each decision in your understanding of the underlying issues at play.

Having the right frame of mind about approaching the problems you are facing matters most. Every other aspect of the user experience development process can be adjusted to fit the time, money, and people at your disposal. No time to gather market research data on your audience? Maybe you can find ways to look at the information you already have, such as server logs or feedback e-mails, to get a sense of the needs of your users. Can't afford to rent a usability test lab? Recruit friends, family, or co-workers to participate in informal testing.

The worst mistake you can make is to gloss over the fundamental user experience issues of the project in the name of saving time or money. On some projects, someone will thoughtfully tack on some form of user experience evaluation to the very end of the process—long after the time to actually address those issues has run out. Racing toward launch without looking back might have seemed like a good idea back when the launch date was set, but the result is likely to be a site that meets all the technical requirements for the project but doesn't work for your users. Even worse, by tacking user experience evaluation on at the end, you might end up launching a site that you know is broken but have no opportunity (or money left) to fix.

Some organizations favor this approach, calling it "user acceptance testing." The word "acceptance" is very telling here—the question is not whether they like the site or will use the site, but rather can they accept the site? This type of testing all too often happens at the very end of the process, by which time countless assumptions have gone into shaping the user experience without ever being examined. Those assumptions can be extraordinarily difficult to uncover in user testing, because they are hidden behind layers of interface and interaction.

Many people advocate user testing as the primary means of ensuring a good user experience. This line of thinking seems to be that you should make something, put it in front of some people to see how they like it, and then fix whatever they complained about. But testing is never a substitute for a thoughtful, informed user experience design process.



Questions that focus on specific elements of the user experience can help you gather more relevant input from your users. User tests constructed without an eye toward the elements of user experience can end up asking the wrong questions, which in turn can lead to the wrong answers. For example, when testing prototypes, knowing what problem you're setting out to investigate is crucial to presenting your test subjects with an experience that doesn't cloud the matter with unrelated issues. Is the problem with that navigation bar really just the color? Or is it the wording that your users are responding to?

You simply cannot depend on your users to articulate their needs. The challenge in creating any user experience is to understand the needs of the users better than they understand those needs themselves. Testing can help you understand the needs of your users, but it's just one of many tools that can help achieve the same end.

## The Marathon and the Sprint

Just as you shouldn't leave any aspect of the user experience to chance, you shouldn't leave your own development process to chance either. Too many Web development teams operate in a state of permanent emergency. Each project is conceived as the response to some perceived crisis, and as a result, every project is behind schedule before it even begins.

I have a metaphor for the user experience development process that I often use when describing problems to clients: A marathon is not a sprint. Know which kind of race you're in and run accordingly.

A sprint is a short race. Sprinters have to call upon vast reserves of energy at the instant the starting gun is fired—and they expend all of that energy in the space of a few minutes. Right off the starting line, the sprinter has to run as fast as he can and keep running as fast as he can until he reaches the finish line.

A marathon is a long race. Marathon runners need just as much energy as sprinters do, but the way they expend it is very different. Success in the marathon depends on how effectively the runners pace themselves. All other factors being equal, the runner who knows when to speed up and when to slow down is far more likely to win—or even to finish the race at all.

The sprint strategy—run as fast as you can from beginning to end—can appear to be the only sensible approach to a race. It seems like you ought to be able to run a marathon as if it were a series of sprints—but it doesn't work that way. Part of the reason it doesn't work that way is the simple physical limit of human endurance. There's another factor here, too: To accommodate that limit, the marathon runner is constantly monitoring his own performance, watching for what's working and what isn't working, and adjusting his approach accordingly.

Web development is rarely a sprint. More often, there will be times when you push forward, building prototypes and generating ideas, followed by times when you pull back, testing what you've built, seeing how the pieces fit together, and refining the big picture for the project. Some tasks are best undertaken with an emphasis on speed; others require a more deliberate approach. Good marathon runners know which is which—so should you.



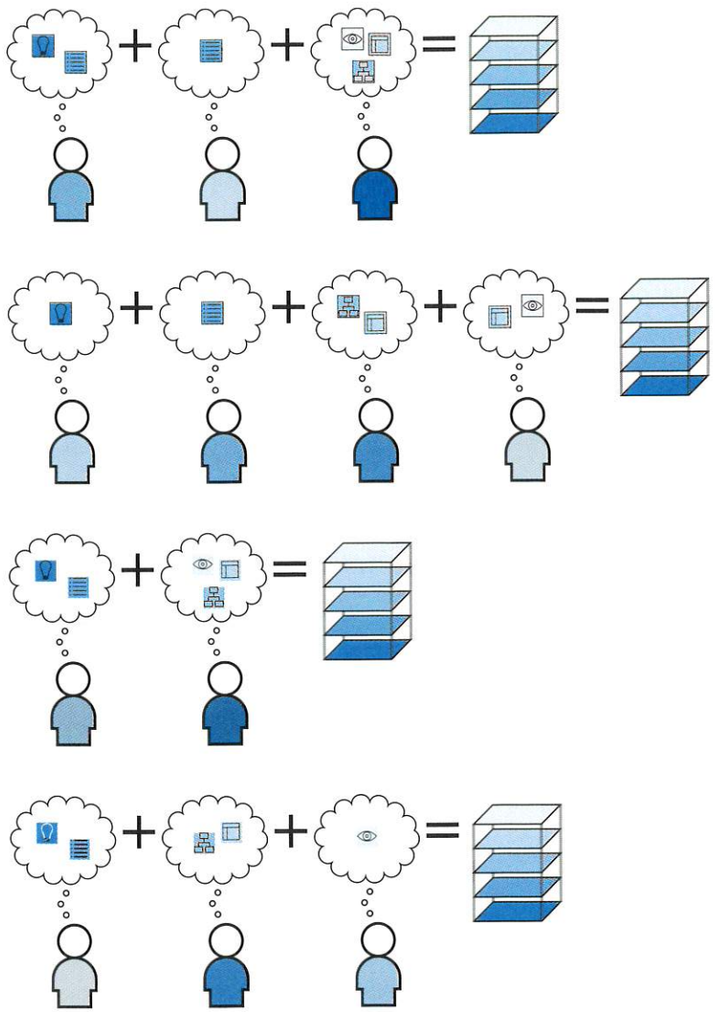
Thoughtful, deliberate design decisions will cost you time in the short term, but they will save you much more time in the long term. Designers and developers often lament the lack of attention to strategy, scope, and structure in the projects they work on. I have been involved in more than one project in which these activities were constantly under threat of being eliminated. Some people get impatient with tasks that don't involve the production of an actual site component like a graphic or a piece of code. These tasks are often the first line items cut from a project that's behind schedule or over budget.

But these tasks were included in the project scope in the first place because they served as essential preparation for later deliverables to come. When they are eliminated, the tasks and deliverables left in the project schedule feel uninformed by the larger context of the project, seeming disconnected from one another.

When you get to the end, you've got a product that falls short of everyone's expectations. Not only have you failed to solve your original problem, you've actually created new problems for yourself because now the next big project on the horizon is to attempt to address the shortcomings of the last project. And so the cycle repeats.

Looking at a site from the outside—or coming into the Web development process for the first time—it's easy to focus on the more obvious elements near the top of the five-plane model at the expense of those closer to the bottom. The irony, however, is that the elements that require close examination to perceive—the strategy, scope, and structure of the site—play a necessary role in the overall success or failure of the user experience.

Only by having someone in your organization think about each of the five planes can you address all the considerations crucial to creating a successful user experience. How these responsibilities are distributed in your organization isn't as important as making sure all the elements of user experience are accounted for.





In many cases, failures on upper planes can obscure successes on lower planes. Problems with visual design—layouts that seem cluttered or busy, or colors that are inconsistent or clashing—can turn users off so quickly that they never discover all the smart choices you made with navigation or interaction design. Poorly conceived navigation design approaches can make all your work to create a sound, flexible information architecture seem like a waste of time.

Similarly, making all the right decisions on the upper planes means nothing if those decisions are founded on bad choices made on the lower planes. The history of the Web is strewn with sites that failed because, although they were beautiful, they were utterly unusable. Focusing on visual design to the exclusion of the other elements of user experience drove more than one startup into bankruptcy and led other companies to wonder why they were bothering with the Web at all.

It doesn't have to be that way. If you approach your Web development process with the complete user experience in mind, you can come out of it with a site that's an asset, not a liability. By making everything the user experiences on your site the result of a conscious, explicit decision, you can ensure that the site works to fulfill both your strategic goals and the needs of your users.