



Devilish Details: Best Practices in Web Design

Larry L. Constantine
University of Technology, Sydney
Constantine & Lockwood, Ltd.

Abstract

Visual and interaction design for successful e-commerce Web sites and Web-based applications requires meticulous attention to detail. Because the smallest matters can ruin the user experience, an orderly process—such as usage-centered design—guided by robust principles is needed; iterative testing and repetitive redesign is inadequate to find and address all the diverse matters needing attention. This paper reviews basic principles and then surveys best practices in the detailed aspects of Web design in three broad areas: details of architecture or organization, details of interaction design, and details relating to commercial activity, especially shopping. Specific recommendations in each area are offered as examples of best practices based on usage-centered principles.

[Keywords: Web applications, e-commerce, visual and interaction design, user interface design, usage-centered design]

ADDITIONAL RESOURCES
(Click items below for more information.)
Training in usage-centered design.
forUSE 2002 Conference Proceedings.
forUSE Newsletter.
Articles and papers.
<http://www.forUse.com>.

[W01]

Devilish Details: Best Practices in Web Design

Larry Constantine

Introduction

The devil dwells in the details. The late Steven J. Gould, with his witty penchant for being simultaneously profound and provocative, preferred to turn the old saw on its head by saying that God dwells in the details. Business-to-consumer electronic commerce on the Web, long the poster child of the Internet age, is living—or electronic—proof that both perspectives can have a certain validity. On the one hand, e-commerce is heralded as the harbinger of a total transformation in how consumers connect with goods and services; on the other, it is often handicapped in meeting even modest expectations by desultory design and inept implementation.

When it comes to design for the Web, the salvation of business success or the damnation of bankruptcy often lies in those blessed and cursed details. Moreover, as Lucy Lockwood, co-inventor of usage-centered design, often admonishes her students, good design is much more than just details—it's details, details, details! Getting all those details right means carefully thinking through myriad seemingly small matters,

always maintaining the perspective of the user and the user's intentions. What hope is there, then for the poor souls designing for the Web?


The Usage-Centered Way

While it is certainly a sin to ignore your users altogether, as one of the lead apostles for usage-centered design, I would argue that part of the problem in e-business on the Web lies in too much attention to users and not enough to what they are doing or trying to do. Lucy and I have sometimes been criticized for making too much of the distinction, but we believe there are differences between user-centered design and usage-centered design, and these differences, even if only a matter of degree or emphasis, do matter. Table 1, adapted from a paper on Web-based applications (Constantine and Lockwood, 2002), clarifies the main differences in philosophy and practice.

Table 1 – Comparison of User-Centered and Usage-Centered Design

User-Centered Design	Usage-Centered Design
Focus is on users: user experience and user satisfaction	Focus is on usage: improved tools supporting task accomplishment
Driven by user input	Driven by models and modeling
Substantial user involvement User studies Participatory design User feedback User testing	Selective user involvement Explorative modeling Model validation Usability inspections
Design by iterative prototyping, trial-and-error, evolution	Design by modeling, engineering process, derivation

What it boils down to is whether you think the path of righteousness lies primarily in communion with your users or in following first principles and disciplined practices. In *user-centered design*, you design something, put it in front of users, get their input, re-design it, try again, build something, test it with users, change it, release it, get feedback, re-design, put it out again—and so it goes. The process, however glorified and codified it may be, ultimately rests on trial-and-error evolution, and testing, in one form or another, is the cornerstone.



In *usage-centered design*, the idea is to get it essentially right by design right from the start. The approach to this Internet nirvana was mapped out in a half century of the quality movement, which recognized that you cannot test your way to quality. Testing comes too late and is inherently both inefficient and incomplete. You reduce manufacturing defects far more efficiently with better manufacturing process up front than with better testing. You reduce usability defects in Web interfaces far more efficiently with better design up front than with testing and refinement later when the best you can do is find out where you went wrong.

That said, you should do usability testing anyway, because even with the best design process, you cannot possibly get everything right. There are just too many details to consider.

The twin pillars upon which usage-centered approach is built are process and principles. First, the approach defines an orderly and efficient process by which the designer proceeds from problem to solution. Second, the process is guided by principles of what constitutes good design, design that is most likely to enable users to accomplish what they intend.

Traditional user-centered design for the Web is a bit like finding your way to a distant destination by starting in some direction and asking the people you meet along the way which way to turn and what road to take. The usage-centered design process is a map of how to get from here to there, and design principles are the signposts along the way. Just follow the map and read the signposts, and you will have a good chance of getting where you are going more or less directly. Neither maps nor signs guarantee that you will not get lost or that you will reach your destination without any detours or backtracking, but, taken together, the process and the principles improve the odds.

The process of usage-centered design.

Here I will give only the briefest of outlines of the usage-centered design process. Other sources (Constantine and Lockwood, 1999; 2002) can fill in the details for those wanting more.

Usage-centered design is a model-driven engineering process based on three simple models: a user model, a task model, and an interface model. But, you say, do not all user interface designers believe in such models? In theory, yes, we would agree, and in practice, perhaps, but the

difference that defeats the devil lies in the particular models by which users, tasks, and interfaces are represented and understood.


Usage-centered design employs abstract models that have been finely honed through experience to capture and clarify the essence of users, tasks, and interfaces in the most expeditious and efficient manner. Regarding users, we are interested solely in the roles they play in relation to a system, and we capture the salient and significant aspects of these relationships in the form of an abstract user role model. For a task model, we turn to task cases: use cases defined by abstract, generalized, technology-free descriptions (Constantine and Lockwood, 1999; 2001). For the interface itself, we begin with simple models of interface contents and maps of navigation paths or other forms of abstract prototypes (Constantine, 1998) to help us get the structure and organization right before we become buried in the details of the real user interface.

By relying on abstract models, usage-centered design avoids being seduced and led astray by distracting details. Of course, for everything there is a time, and the time to become immersed in details is after you have explored and fully understood the territory; detailed and local decisions come in their turn once general and global issues have been truly resolved. For holding the devilish details at bay until you are fully equipped to deal with them, the pay off is enormous. Abstraction works, leading to genuine innovation and truly world-class results (Constantine and Lockwood, 2002a; Constantine and Windl, 2001; Constantine, 2002; Windl, 2002).

The principles of usage-centered design.

If you believe that interaction truth and interface beauty lie only in the eyes of your users, you are condemned to rely on them for revelation. Only feedback from users, testing with users, and usage by users will reveal the details that mark the path out of the Web-business wilderness. On the other hand, if you believe that usability is based on discernible and recognizable principles of good form and interaction, then it is possible, however difficult it may seem, to make your way more or less directly to good design.

Although we often focus more on process than principles, design principles undergird all our work. Indeed, we might just as well have called our approach principle-based design. Before we had a process in which to



apply them, we had cataloged and were using a coherent set of design principles.

Many long lists of design principles and rules have been compiled. Because we are interested in efficiency and simplicity, our list is shorter than most, compressing the guidance that others spread among hundreds of hard-to-remember rules into only a half dozen broadly focused design principles: visibility, feedback, structure, reuse, tolerance, and simplicity.

Visibility is about showing or offering users whatever they need for the task at hand without distracting or overwhelming them. Burying crucial functions and exposing gratuitous features are equal sins against this principle.

Feedback is about keeping users informed about and in terms of what is relevant to them in the task at hand. Failing to notify users that your programming could not make sense of some input is as bad as annoying them with needless confirmations of their every small step.

Structure is about deliberate organization that reflects the intrinsic or familiar organization of things. Things that are similar or related ought to resemble each other; things that are not, ought not.

Reuse is about recycling visual elements and interaction patterns so that interfaces are not only consistent, but contain fewer distinct things to understand and master.

Tolerance is about being flexible and forgiving in interaction with users. Users should be helped to make fewer mistakes and aided in correcting or overcoming mistakes they do make.

Simplicity is about genuine elegance and parsimony rather than simple-minded reductionism. Shoehorning all functions of a set-top remote into a single combined rocker-push-button control for navigating a many-leveled on-screen menu is stupidity rather than simplicity.

What's to argue with? The ingredients in this list are certainly not original; in one form or another they are probably already familiar to most usability professionals.

Details, Details, Details

By keeping these principles in mind it is possible to anticipate and avoid in the first place many of the mistakes in detail-level design that are otherwise not uncovered until usability tests are performed and analyzed or until hoards of customers click their way to your competitors' sites.

For example, one e-commerce site trumpeted how, after losing millions in sales because customers were bailing out from the checkout process, extensive user testing enabled them to uncover the cause. The validation of expiration date had been written to expect all digits. Customers who entered dates as these appear on most cards—for example, “04/03”—were informed that their credit card information was invalid and were told to enter the details again.

You do not have to lose sales or spend time and money on user testing to get this one right. You only have to remember the principles of tolerance and of structure. First, tolerant design accepts whatever users type and guides them in changing what cannot be interpreted by the programming. Second, the structure of the data within the form should reflect the structure on the card. This is not only more polite, but it reduces transcription errors, which are costly to all concerned.

If your programmers cannot parse a date with slashes or dashes included, fire them. If your designers do not include templates (“Date (mmdd):”) or examples (“E.g., enter April 2003 as 04/03.”) when needed, train them or get rid of them. If no one can write informative error messages or screens that politely inform the user what the site could not understand and why, then get a new process.


While we are on the credit card form, the same principles can be invoked to resolve another design detail. For successful purchase, the user must correctly enter the credit card number as well as the expiration date. Since a majority of consumers possess more than one card, the odds are good that the user is looking at the card when entering the details.

With very few exceptions, credit card numbers around the world consist of a string of 16 digits arranged in groups of four. This structure not only has significance to the banks and credit card processors, but it makes it easier to parse the number visually, read it aloud, and verify it. When copying a card number into a Web form, it is natural for the user to type something like:

4445 5666 6777 8999

Perversely, many sites reject this and give the customer an error message, sometimes informative and sometimes not. (I actually got one that read simply “Invalid card number.”) The user who persists discovers that the number must be entered as a contiguous string of digits, thus:

4445556667778999



This format increases the chances of a transcription error in the first place because users are not typing what they see. In the second place, it makes it visually verifying the input as typed on the screen against the number on the actual card much more difficult and error prone. There is, for example, a mistake in the second number above that would be much easier to spot were it to include the spaces. Worst of all are those forms that use asterisk-fill password-style entry for credit card numbers, making them impossible to verify visually.

Do such small details matter? Yes, to the tune of millions of dollars of incomplete or incorrect retail transactions. Sites that ignore these principles in designing checkout facilities can expect to have more abandoned shopping carts and more incorrect credit card numbers.

Toward Best Practices

Countless other examples could be considered. Because there are so many, this paper will focus on those issues of design detail that are either particular to design for the Web or of special salience in ecommerce. Because the messy details of successful design are necessarily messy, it is all but impossible to explore them in a truly logical order. This review of best practices in design for the Web will be organized into three loosely defined and possibly overlapping areas:

- Details of architecture (if that is not a contradiction in terms).
- Details of interaction design.
- Details relating to commercial activity, especially shopping.

Within each of these areas, a representative sampling of issues will be explored with no pretense of encyclopedic treatment.

Architecture Click by Click

Branching logic

An all-too-common practice on the Web is to offer navigation that attempts to categorize customers and channel them to a particular sub-site tailored to their needs. While seeming to be a good marketing technique that also helps create a better user experience, the ploy often fails in practice, particularly when the alternatives offered are confusing to customers or make too many assumptions about what a particular kind of customer will want or will be allowed to see or buy.

A common example of branching logic is the distinction between home, home business, small business, and large business. The boundary between small and large is fuzzy, and even when the categories are spelled out (“Large Business – Over 100 employees”) customers near the cut-off point may agonize over the choice.


One of the most frustrating experiences to users is being certain that something is on a site or in an application from having seen it before, but being unable to find it again. The chances of this happening are increased with deep information hierarchies arranged with strict branching logic down mutually exclusive paths.

For example, Dell.com offers (or forces!) a selection within major user categories: Consumer, Business, and Public. Among others, these options include: Home & Home Office, Small Business, and Medium & Large Business. Right there we have a problem. What if you are a small business operating as a virtual corporation out of a home office? Even if your business category is unambiguous, what does it mean to choose one route over the other. Do you get more information if you go down one hallway? Better prices? More options?

As it turns out, the Home & Home Office branch and the Small Business branch are organized so differently that one imagines they are being designed and managed by completely different business units that do not even talk with each other. Things that are easily found down one path are difficult or impossible to reach going down the other. As it turns out, the links provided by Dell on my laptop and my desktop machine are set differently, so that I once spent a frustrating hour trying to find and order something from my office after I had already checked out the product while on the road.

The branching illogic on this site continues deep into the site. One gets a different list of available monitors depending on which branch is taken, whether you follow links or do a search, etc. In practice, it is a site within which it is easy for the user to get lost and to lose track of information.

Branching logic that takes users down mutually exclusive paths is often problematic. In general, users have a better chance of finding what they seek if there are multiple paths to the same target and if the target is the same regardless by which path it was reached. A PC marketer may want to lead with certain simple configurations for consumers, but the same option to “View all models” should appear prominently at the same point whether the consumer path or commercial path is taken. Because users might go down more than one path in search of a particular product



or combination of options, the same basic options and information should have the same basic appearance and behavior regardless of how approached or reached. The overall visual configuration of a page is one way users know and keep track of where they are. (“I’ll recognize that page when I see it again.” “I don’t recognize this! Where are the rest of the models?”) In contrast, at dell.com, apparently meaningless visual differences abound. Compare, for instance, three alternative paths to “notebooks” from the dell.com home page:

http://www.dell.com/us/en/bsd/products/line_notebooks.htm

http://www.dell.com/us/en/dhs/products/series_inspn_notebooks.htm

http://www.dell.com/us/en/gen/products/line_notebooks.htm

Information architecture and product hierarchies.

Information architecture is a Web-born neologism with varied definitions and interpretations but encompassing elements of presentation, database organization, usability, visual and interaction design, and even aspects of graphic design. The primary focus is on how a site and the information it encompasses are organized or structured and how this organization is conveyed to users.

One aspect of information architecture is particularly important for e-commerce: how the entire line of product offerings on a site is organized—not only as presented to the customer but as structured as a product hierarchy on the back end and in the rest of the business. Details of such organization and even interactions between different structures can significantly affect usability and the likelihood of customer success. After countless frustrating experiences, I have concluded that many sites do little more than take the internal business organization or the structure of back-end databases throw it at the users. A product hierarchy that makes complete sense to the business or its database applications can be utter nonsense to customers.

For example, if I am seeking clip-on conference badges with inserts that can be printed on a laser printer, where do I look on Staples.com?

The Staples.com site (see Figure 1) gets many details covered in this paper right: a persistent shopping can be seen to the upper right, tabs across the top are replicated by text links within the center channel and at the bottom, and more. However, although the home page gives the impression that I should be able to readily find what I need either by

searching or by following links, due to interactions between an ineffective search facility and a confusing product hierarchy, this is not the case.

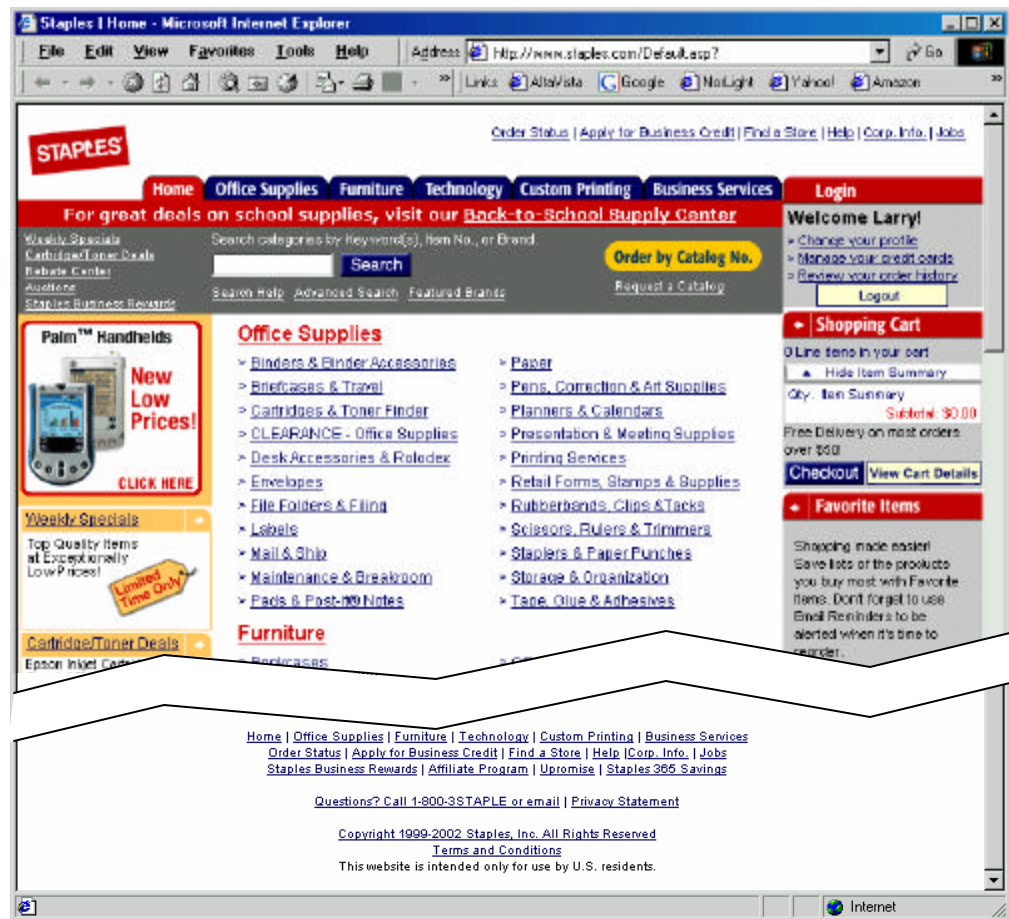



Figure 1 – Staples.com home page, above the fold and at the bottom.

Starting with an obvious search on “clip-on conference badges” returns a strange result page with 8 randomly irrelevant categories ranging from furniture to phone systems and time cards plus one promising one: “Office Supplies / Labels / Cards & Badges / Name Badges.” Clicking on “Show all 6 matching items” does not lead to the needed product. Perhaps I had over-specified my search. Mysteriously, searching on just “conference badges” returns only one hit, which is not the product I want.

At this point, many consumers would give up, but I needed the badges and was certain Staples carried them. Besides, I am a usability



professional and wanted to figure out what was going wrong. From the home page, one promising link was: “Office Supplies / Presentation & Meeting Supplies.” However, only by process of elimination could I pick “Meeting Room Supplies” from among links on the next page. That took me down another level in the hierarchy to where “Name Badges” is among the three links. Click again and still no luck, although closely related products were listed.

A search on “name badges” revealed a completely different route through the hierarchy leading to different products: “Office Supplies / Labels / Cards & Badges / Name Badges.” How many consumers would think of cards and badges under the category of labels. However, as they say in Maine, “You can’t get there from here.” Actually following these links from the home page does not lead directly to the product list, but only to another set of what Jared Spool calls “category links,” one of which (“Avery Laser/Ink Jet Name Tag Kits and Refills”) yields a page topped by a big and uninformative graphic that buries the right product below the fold. I could not have known in advance that what I wanted was Avery CC104C Laser/Inkjet Name Badges, Clip Style Kit, 3" x 4".

Been searching.

Search facilities present a genuine design dilemma. On the one hand, many regular Web surfers insist on having ready access to a search box. On the other hand, results typically resemble the frustrating experience with Staples.com described above.

The ugly but hidden truth of ecommerce is that searching does not work. Studies confirm that the chances of success are cut substantially if a visitor uses the search function. In fact, many users only turn to the search function when scanning for and following links fails. Good visual design and sound navigation schemes can obviate the need for searching.

The problem is that ordinary consumers do not have the advanced skills needed to formulate well-formed queries or devise effective search strategies. There is a mismatch between the logic of ordinary users and that used by the vast majority of search schemes, as evidenced by my failed attempt to broaden the search on Staples.com. Search portals like Google have demonstrated that it is possible to design search facilities that work for the user more often than not, but searching is their business, not one small piece in a retail sales and fulfillment site. Least useful of all are generic schemes that do exact-match full-text searching.

The Staples.com example points the way to some crucial details to be considered in the details at the intersection of searching and structure:

- The same category must contain the same items, however it is reached.
- Search engines must recognize alternative terms and equivalences (“conference badges” and “name badges”) based on a rich and custom-tailored glossary.
- Site design should favor or promote alternatives to searching, such as rich category structures, site maps, and site indexes.
- Category listings should include “see also” or “related product” listings. For best results these may need to be tailored or tweaked by hand.
- The terms used in navigation links, category listings, maps, and indexes must be in the customer vocabulary, not in the industry terminology or the categorization within the database schema.

Navigation, access, and place.

The goal in designing navigation schemes for Web ebusiness is to maximize the chance that users will find what they seek. Redundancy in the paths to any given target speeds and simplifies use and improves the chances of success.

For example, primary navigation links should be replicated in both text and graphic form, with a redundant set of text links at the bottom of every page. (Staples.com gets kudos for this.) Many power surfers have a penchant for going straight to the bottom of a page in search of useful links. Users who have more casually scrolled or browsed their way to the bottom of the page will not be lost or forced to return to the top of the page in order to take the next tack. For the same reasons, wherever forward and back or next and previous links/buttons make sense in terms of the structure of the information, these should be provided at both the top and bottom of pages or forms.

Similarly, within long pages, key navigation can be repeated within long pages, as in the example of the index shown in Figure 2, where the alphabetic navigation bar that marks the boundaries between letters, highlights the current letter, and offers immediate access to any letter from any place in the page. Top-of-page links or buttons should also be

replicated at intervals in long pages for those users preferring mouse operation to keyboarding.

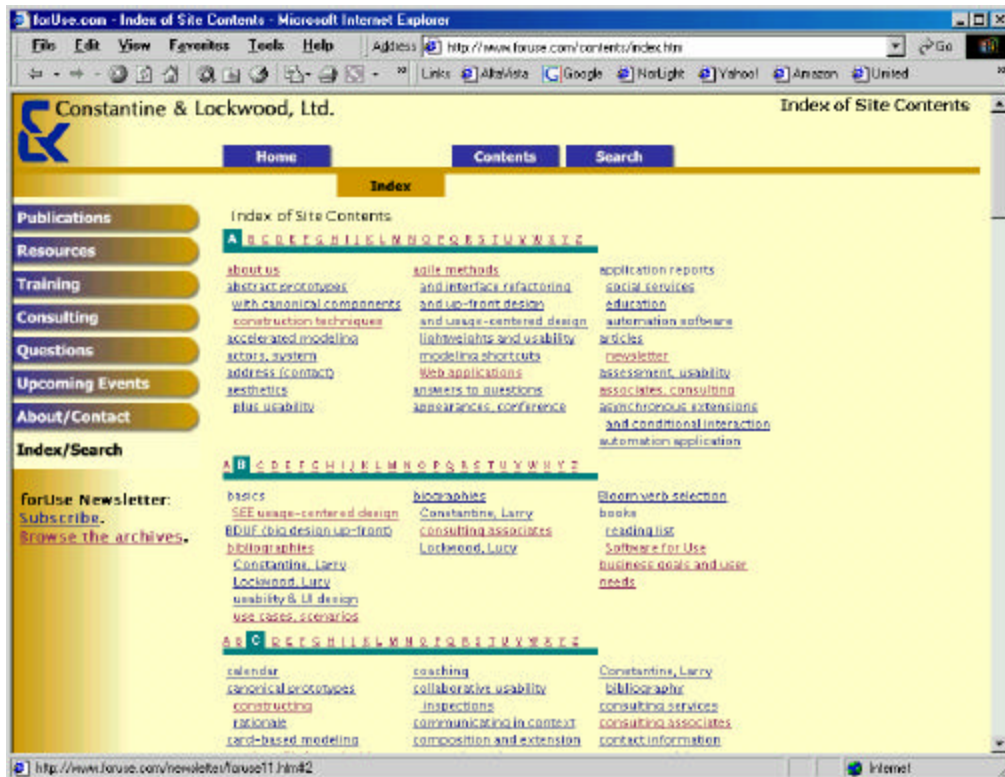


Figure 2 – Index page with repeated alphabetic links.

Maintaining a sense of place on the Web is a challenge for users and for designers. Users must not only immediately recognize where they are on the Web—on your site and not some competitor—but always know where they are within your site. This means that all pages within a site should, in some sense, look alike at the same time that every page is different. These competing objectives have to be met while also satisfying aesthetic aims.

Simply telling users where they are with “breadcrumbs” or other schemes that merely expose the page hierarchy do not typically meet the needs of most users and fail to satisfy the criterion of instant recognition. The minutest details of graphics, layout, text styles, and the use of color and space must be designed to work in combination to show users where they are and where they can go next.

Successful designs are generally based on the technique of theme and variation, with a common look and feel carried across an entire site within which a small number of similar but readily distinguished layouts are presented. A handful of related templates are defined to distinguish classes of pages based on meaning to the user in combination with position in the page hierarchy. The home page must look like a home page and serve as a visual and conceptual anchor for the user. Other important classes include: first-level or primary pages, intermediate-level pages, bottom-level or target pages, documents, message and notification pages, search results pages, indexes, and so forth.

Very small details spell the difference between schemes that work and ones that do not. When we redesigned forUse.com, we defined a two-level core navigation scheme, shown in the design mockup of Figure 3. Primary navigation through horizontal “tabs” in the left channel gave direct access to sections of the site, each with its own collection of from one to five second-level pages. Sections with multiple pages at that level were navigated as a “tab set” using a form of “index tabs” arrayed across the top of the page. The sense of place is maintained and communicated by changes in appearance in the left-channel and top-bar navigation tabs.

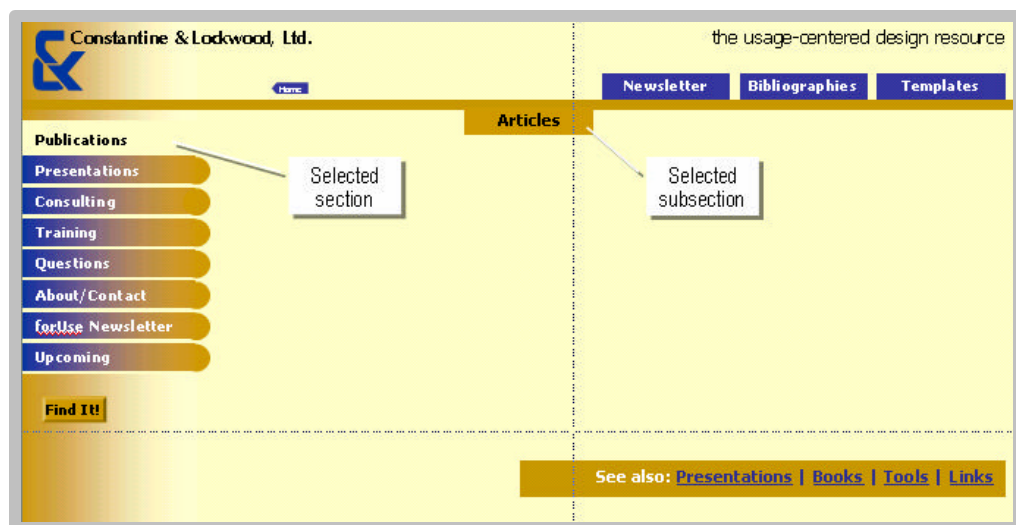



Figure 3 – Mockup of the original design for new forUse.com navigation scheme.

The scheme did not work for several reasons. Despite the horizontal stripe visually anchoring the top tabs to the left channel, users did not see



the connection. Some missed the tabs altogether, others failed to see them as navigation controls, and some were just bewildered by seemingly random changes in appearance. Many criticized the inconsistent style of the “Find It!” button, which had been intended to help distinguish the function from other links, and the tiny “Home” button, which was intended to reinforce the logo as a link to the home page.

The original idea was basically sound but was defeated by subtle mistakes in the realization. The final scheme, which can be seen in Figure 2, differs only in minute but important details worked out through agonizing attention in redesign. All navigation tabs have been given drop-shadows to help communicate affordance as active controls. Affordance is reinforced by mouse-over effects. Left justifying the top-of-page tabs and including an identical “Home” tab in each set was sufficient to reinforce the visual and functional connection with the left-channel tabs. The anomalous “Find It!” tab was folded into the main navigation set as an “Index/Search” tab.

Interactive Forms

When it comes to the humble HTML form, design details loom large indeed. Forms are the plow-horse pages of the Web. They do most of the work and bear the brunt of problems in user interaction.

Mistakes were made.

Users will make mistakes, and in most but not all cases, it will be in their interest and yours to catch mistakes as early as possible. Once detected, errors in forms are best reported in context, that is, within the same page where they are found. With or without a separate error message, the specific fields should be highlighted or marked for quick location and recognition using color reinforced by text, texture, graphics, line weight, or other means not dependent on color discrimination.

If a separate confirmation page is presented, best practice would allow users to make changes or edit in place without having to return to the prior page. In some cases, allowable changes may have to be restricted to a limited set of options, such as canceling an already entered order.

Contingency design (Linderman, 2002) is “design for when things go wrong.” It assumes that mistakes will be made, that exceptions will occur, and that designers should design for them. One trick is to identify and design for the most frequent or common mistakes, such as typical

misspellings of product or brand names in searches. Sites that collect and analyze search terms and click-through paths will be in a better position to design around such common user errors.

In many cases, validation schemes for form fields need to be tempered by tolerance. For example, telephone numbers are numbers, but they often include non-numeric characters, and these characters vary from user to user and country to country. Although telephone numbers have structure, it may be best simply to strip all non-numeric characters. If a particular format is required for subsequent processing, a template can be displayed (for example, “country code (city/area code)-number”) or separate fields presented for each part of the number.

As in the earlier example, credit card entry should accept spaces. An example can even encourage users to enter in groups, but requiring groups of four digits separated by spaces is probably not a good idea.

Submit gracefully.

In a sense, the Submit button is what makes e-business possible. Without it, customer orders would never be received, feedback would never arrive, and credit cards would never be charged. However, the Submit button is also frequent point of failure in ecommerce that leads to perplexed or irate customers and lost sales.


In its standard form, it looks like this in HTML:

```
<input type="submit" value="Submit" />
```

The problems begin with the label on the Submit button, which HTML in its perversity calls its value. On all too many sites the button just reads “Submit” when to the user it really should say “Order Now” or “Bid” or “Buy Tickets.” The button should say what it means to the user. In addition, a pop-up screen tip should give the hesitant user an informative and reassuring explanation.

Problems with submitting forms continue in the interaction design. The default behavior of the standard HTML Submit button has several significant usability defects. First, every time the user clicks on Submit, a new set of data is sent to the CGI script that processes the form. In case the problem is not abundantly clear, consider the following all-too-familiar sort of message.

IMPORTANT: Click on the Submit button ONLY ONCE. Please wait. Processing may take several minutes. Do NOT click the



Submit button again or you may receive and/or be charged for duplicate orders.

Why is the user being told what not to do when it's the program's responsibility? In one suite of commercially available JavaScript widgets, the user who clicks twice gets an alert box, the one who clicks yet again, gets an alert box with the message:

What part of once don't you understand?

Nice manners, eh?

Another problem with the standard Submit button is that it intercepts the <Enter> key. A user who ends a text entry by hitting <Enter> may find the form has vanished only to be replaced by an error message that it was incomplete. The programmer-oriented solution to this, which has actually been packaged as a standard component for sale, is a JavaScript that intercepts the user action and asks:

Did you really want to submit the form?

How, then, should a Submit button behave if it is designed for use? What does the user expect?

First, if double-clicks are a problem, the Submit button should be disabled the moment it is clicked. Because users will not necessarily know what is happening, they should be informed that their form—whatever it may be—is being processed and they need to wait. The form should display a message along the lines of:

Please wait while we process your order.

A pop-up message box that requires confirmation is NOT the way to deliver the information; it simply annoys users and imposes another action that to users is unnecessary. As in many cases, the place to inform the user is in context, near the button that triggered the forms processing.

As it turns out, the programming to correct both these problems—double-clicking and unintentional use of the <Enter> key—in the same button is so straightforward that one wonders why Web developers continue to deliver faulty forms or lame fixes. A small part of the fault lies with lazy programming, with making casual use of the default HTML Submit button without even so much as a thought about its appropriateness to an application. However, the greater part of the fault lies with usability professionals for failing to pay attention to such details and for not telling our programmers that these things matter. If we pay

attention to details and talk with our programmers, they will write something like this:

```
<input type="button" value="Bid!" name="Submit"
title="Submit this bid in auction." tabindex="6"
onClick="javascript:validateForm();">
```


To summarize the best practice in forms details:

- Form submit buttons should have functional labels that are meaningful to users.
- Screen tips (as titles or alt-text) should offer added explanation.
- The button should become disabled on click and re-enabled on any change in the form.
- Once clicked, the user should be informed and reassured by a message that processing has started.
- An in-context message is better than a separate message page that makes the user's information vanish.
- Simple animation works best for conveying the sense of processing progress. and be.
- For longer forms, especially where the Submit button may fall below the fold, the <Enter> key should not trigger submitting the form.
- For search boxes or very short forms with one or two fields, the <Enter> key should be enabled as an alternative to clicking on the Submit button.

In default.

Choosing the right default value for check boxes, option buttons, and pre-filled text boxes can substantially increase efficiency and improve the user experience. Default values must be carefully chosen based on the particular situation. For example, designers should not hesitate to leave all options unselected in a set of radio buttons when it is necessary to guarantee the user has made a deliberate selection.

Default values should reflect information about the user and be dynamically updated to reflect prior input. For example, if a known user is logged in on a travel site, the departure city can be pre-loaded appropriately. Once a user has selected a departure date for a trip, the return date should be advanced to the following day.



Remember, too, that some users prefer to navigate through forms by keyboard, so tab order must be carefully planned.

Offering help.

Although relatively uncommon on the Web, providing useful and usable help facilities is a best practice for e-business sites as well as Web-based applications.

Instructive interaction (Constantine and Lockwood, 2002b) is an approach that facilitates first-time use without separate help or instruction. The varied techniques of instructive interaction are particularly appropriate for e-commerce on the Web. Help should be offered in context through examples, templates, brief prompts, screen tips, and embedded prompts. Embedded prompts—in which a short prompt appears within a text-box or drop-down but disappears when the object receives focus—are becoming increasingly common in forms on the Web.

Immediate access to closely related information can spell the difference between a sale and a lost customer. For example, making size charts and definitions available whenever a size is indicated can reduce returns by helping customers make the right choice.

The presence of required field and their styling should always be noted in advance. Although it is common to flag required fields with an asterisk, this should be noted at the top of the form, not as a footnote at the bottom. Redundant flagging of required fields—with a special character or glyph plus bolding, highlighting, outlining, or a distinct color—is the best practice.

Shopping

They call it a shopping cart but it does not behave like one. In the real world, shopping carts do not go away to be stored in another room, and they do not require reducing a quantity to zero to remove an item. Shopping carts on the Web should be persistent, that is, they should follow the user around the site and remain visible with visible contents in the form of a list of items or at least a count. This best practice, conceived years ago (Constantine, 2000), is finally becoming more common on the Web.

Putting something into a shopping cart ought to be the easiest thing in the world, yet it is surprising how many sites do not provide the necessary button or link in every place. One place often missed is in search results

lists. Customers should not be required to drill down in order to make a purchase.

As soon as customers have enough information for a decision, they should be able to buy something. For that reason, surfacing more information about products, especially in search results and product listings, is another best practice. Instead of just naming items, lists should include as much information as possible to avoid sending users bouncing around between pages in search of the right product.

Based on real-world shopping behavior in bricks-and-mortar stores, Web sites should support a wide variety of task cases related to customers looking, finding, and buying. Among these task cases are:

- finding specific known product(s)
- finding specific product(s) by type/category
- finding specific product(s) by function/use
- finding specific product(s) by description
- finding similar/related product(s)
- looking/browsing for specific product(s)

As the Staples.com example illustrates, even the first of these, simple though it may appear, can be a challenge for the customer to complete successfully in some cases. Other common task cases that are vital to shopping are not usually well supported on the Web. These include:

- finding specific product(s) by appearance
- looking/browsing for something
- looking/browsing for what's available
- comparing alternative products
- checking availability
- checking current total cost/shipping cost

Sites aiming for the best user experience and highest conversion rate need to support these task cases in simple and user-efficient ways. Current total, including shipping, can be continuously updated and displayed in the persistent shopping cart. Availability should be noted as part of any display that includes a product, without forcing users to drill down to the detail view.

Technology steps.

Some parts of the typical Web checkout process are particularly annoying examples of what we call “technology steps”—actions that are not part of the user’s agenda but are present because of the failure to overcome or work around limitations in the technology. The “recalculate total” or “update order” button is a common example. Recalculating the total is not the user’s problem; it should happen automatically anytime the form has been changed. There are dozens of ways to achieve this behavior on the Web.

Perhaps the worse sin in the form of a technology step is allowing user-entered data to vanish through use of the back button. Never lose, discard, or corrupt user input. Requiring the user to re-enter the information in a long form is one of the surest ways to lose a sale and possibly a customer.

Another technology step that mars the final stages of the purchasing process is the need to change the quantity to zero to remove an item from the shopping cart. Best practice provides a button or glyph to remove an item.

You Can’t Do That

It is worth noting that when I first designed a persistent shopping cart with automatic update, some programmers told me it could not be done on the Web and were eagerly explaining why it was impossible at the same time that others quietly set about programming it. Regardless of what your programmers claim, every example offered in this paper is realizable. Indeed, there is always more than one way. Designers need to have the courage to design to best practices and demand the best from programmers. Essentially anything that can be achieved in a stand-alone or client-server environment can be deployed over the Web and operate within a browser instance.

The usage-centered design approach to best-practice design on the Web can be summarized this way:

- Design it right to fit the users and their intentions based on the assumption that implementation and deployment technology will present no problems.
- Translate this ideal design as literally and directly as possible into one for deployment on the Web.

- Wherever possible, strive to overcome any limitations or inefficiencies due to current technology.
- Compromise as needed—but only as needed and only careful deliberation and full exploration of implementation alternatives.

References

- Constantine, L. L. (1998) Rapid abstract prototyping. *Software Development*, 6 (11), November. Reprinted in S. Ambler and L. Constantine, *The Unified Process Elaboration Phase: Best Practices in Implementing the UP*. Lawrence, Kansas: CMP Books, 2000.
- Constantine, L. L. (1999) Persistent models: models as corporate assets. *Software Development*, 7 (11), November. Reprinted in L. Constantine (ed.) *Beyond Chaos: The Expert Edge in Managing Software Development*. Boston: Addison-Wesley, 2001.
- Constantine, L. L. (2002) Featured design portfolio. *ACM interactions*, 9 (2), March/April.
- Constantine, L. L., and Lockwood, L. A. D. (1999) *Software for use: A practical guide to the models and methods of usage-centered design*. Reading, MA: Addison-Wesley.
- Constantine, L. L., and Lockwood, L. A. D. (2001) Structure and style in use cases for user interfaces. In M. van Harmelan, Ed., *Object Modeling and User Interface Design*. Boston: Addison Wesley.
- Constantine, L. L., and Lockwood, L. A. D. (2002a) Usage-centered engineering for Web applications. *IEEE Software*, 19 (2), March/April, pp 42-50.
- Constantine, L. L., and Lockwood, L. A. D. (2002b) Instructive interaction. *User Experience*, 1 (3), Winter.
- Linderman, M. (2002) Making mistakes well. *new.architect*, 7 (9): 32-34, September.

Get Your Copy Today

forUSE 2002 1st Int'l Conference on Usage-Centered Design Complete Post-Conference Package

- **Conference Proceedings** - professionally edited, perfect-bound volume with 36 papers.
- **Conference CD-ROM** - PowerPoint and Acrobat copies of presentations, plus searchable electronic copy of the Conference Proceedings. Nearly 240 megabytes.
- **Session Materials** - Spiral-bound hardcopy of session handouts.
- **Bonus** - Large conference carry-bag to keep it all in (while supplies last).



Proceedings and CD include important papers and presentations by such recognized experts as **Larry Constantine, Gloria Gery, Karen Holtzblatt, Jim Hobart, Lucy Lockwood, Jeff Patton, and Rebecca Wirfs-Brock**. 36 chapters cover a wide range of vital current topics, including:

- Usage-Centered Design in XP and Agile Development
- Usage-Centered Design and the Rational Unified Process
- Designing Information Portals
- Best Practices in Design for the Web
- Performance Support for Complex Problem-Solving Tasks
- Usage-Centered Exploration to Speed the Initial Design Process
- The Role of Scenarios in Usage-Centered Design
- Web Design Patterns for Transactional Systems
- What It Really Takes to Handle Exceptions in Use Cases
- Designers as Change Agents

\$150 each (plus shipping and handling)
Use order form on next page...

On-line resources at foruse.com

- Over 70 articles, papers, and presentations
- Answers to more than 40 frequently asked questions

forUSE: The Electronic Newsletter of Usage-Centered Design

Join more than 2,000 forward-thinking designers, developers, and managers who subscribe.

- First access to new information and developments
- Special discounts on seminars, conferences, and materials.

Click here to subscribe now at

www.foruse.com/forms/subscribe.htm

ORDER FORM

forUSE 2002 1st Int'l Conference on Usage-Centered Design Complete Post-Conference Package

FAX (credit-card orders) to: **+1 978 948 5036**

or

MAIL (check enclosed) to:

Constantine & Lockwood, Ltd.
58 Kathleen Circle
Rowley, MA 01969

Please send [] copies of the forUSE 2002 Complete Post-Conference Package at \$150 each plus shipping and handling (see below).

Shipping & handling (check one):

- Domestic Surface (\$10 each) Domestic 2nd-Day Air (\$25 each)
 International 1st Class (\$30 each)



Name:

Shipping Address:

Telephone (required):

Email (required):

Payment by (check one): Personal Check (enclosed) Company Check (enclosed)
 Visa MasterCard American Express

Name on card: _____

Card number: _____

Expiration date: _____

Signature: _____