3 A Basic Framework

CHAPTER

Deborah J. Mayhew Deborah J. Mayhew & Associates
Marilyn M. Tremaine New Jersey Institute of Technology

3.1 INTRODUCTION

This chapter presents a framework for cost-justifying usability engineering efforts on software development projects by describing how to *calculate the costs* and *estimate the benefits* of each of the Usability Engineering Lifecycle tasks that can potentially be applied (Mayhew, 1999). We present a general framework that is relevant to developing software for any application from commodity trading to a Web commerce site.

A usability engineering cost-benefit analysis is conducted in the software development process for two major reasons:

1. To demonstrate that usability engineering is a viable and significant cost saving approach
2. To plan the usability engineering program for a particular development project

General cost-benefit analyses of hypothetical usability engineering efforts can be prepared as a strategy to win general support for trying out usability engineering tasks and techniques in a software development organization. When an organization has no experience with usability engineering, cost-benefit analyses may be an effective way to free up resources for usability improvement efforts.

In organizations that are more mature with respect to usability engineering, cost-benefit analyses can be used to plan an optimal usability engineering program for a particular software development project. To help settle on a final usability engineering plan for a specific development project, costs are calcu-

lated for the most aggressive program that can be implemented, including the most reliable and thorough *techniques* for all Lifecycle *tasks* (see later discussion). Then benefits are predicted, using *conservative* predictions. If benefits still outweigh costs dramatically, as they invariably will when critical parameters (e.g., number of users, volume of transactions) are favorable, then even the most aggressive usability engineering program can be argued to be viable. This is because only the most conservative claims concerning potential benefits have been made. In fact, the benefits predictions can be redone using more aggressive but still realistic benefit assumptions. The new calculations will show that, in all likelihood, an even more dramatic benefit can be obtained, even from a significant investment in usability engineering.

If, however, benefits and costs in the initial analysis match up fairly closely, then the initial, aggressive usability engineering plan needs to be scaled back, possibly to even a bare-bones plan. It is still possible to achieve the very conservative benefits with even shortcut usability techniques, and thus predict with confidence a healthy return on investment (ROI) from a minimal approach to usability engineering. It is wiser, in the long run, to engage in a conservative benefit assumption and spend a small amount of money, than to barely achieve predicted benefits with a large expenditure. Large expenditures with little to show for them rapidly destroy a manager's or consultant's credibility.

Thus, cost-benefit analysis can be used to develop a cost-effective usability engineering effort for a software development project that is likely to pay off as predicted.

When an organization is first experimenting with usability engineering techniques and is still skeptical about their value, it is wise to make extremely conservative cost-benefit arguments, based on a relatively low-cost usability engineering effort and very modest predictions of benefits, and then to show, after the fact, that much larger benefits were realized. Once an organization has had several positive experiences with investing in usability engineering, it will be more receptive to more aggressive proposals for usability engineering programs, and also to more optimistic benefits predictions.

3.2 THE USABILITY ENGINEERING LIFECYCLE

The first step in cost-justifying usability engineering on a particular software development project is to lay out a usability engineering plan for that project.

The Usability Engineering Lifecycle (Mayhew, 1999) documents a structured and systematic approach to addressing usability within the product development

process. It consists of a set of usability engineering tasks applied in a particular order at specified points in an overall software development lifecycle. Readers familiar with the Usability Engineering Lifecycle might wish to skip this section, which provides an overview of the Lifecycle. Readers interested in more detail than that provided in this overview are referred to Mayhew (1999).

Several types of tasks are included in the Usability Engineering Lifecycle, as follows:

- Structured usability requirements analysis tasks
- An explicit usability goal setting task, *driven directly from requirements analysis data*
- Tasks supporting a structured, top-down approach to user interface design that is *driven directly from usability goals and requirements data*
- Objective usability evaluation tasks for iterating design towards usability goals

The chart in Figure 3.1 represents, in summary, visual form, the Usability Engineering Lifecycle. The overall Lifecycle is cast in three phases: Requirements Analysis, Design/Testing/Development, and Installation. Specific usability engineering tasks within each phase are presented in boxes, and arrows show the basic order in which tasks should be carried out. Much of the sequencing of tasks is iterative, and the specific places where iterations would most typically occur are illustrated by arrows returning to earlier points in the Lifecycle.

In addition, free-floating text not enclosed in boxes makes general reference to tasks in an underlying software engineering methodology, with which the Usability Engineering Lifecycle tasks—which are the main focus in this chart—would be conducted in parallel and with which they would be integrated. For example, Lifecycle requirements analysis tasks would be conducted in parallel with function and/or data modeling tasks in many software engineering methodologies, and in particular with the development of a "Requirements Model" in the Object Oriented Software Engineering (OOSE) methodology (Jacobson *et al.*, 1992). These notes provide a general idea of how the Usability Engineering Lifecycle must be integrated with an underlying software engineering methodology.

In considering how to adapt the Usability Engineering Lifecycle to a Web development project—or any type of project for that matter—the distinction between *tasks* and *techniques* is an important one.

A usability engineering *task* can be defined as an activity that produces a concrete work product that is a prerequisite for subsequent usability engineering

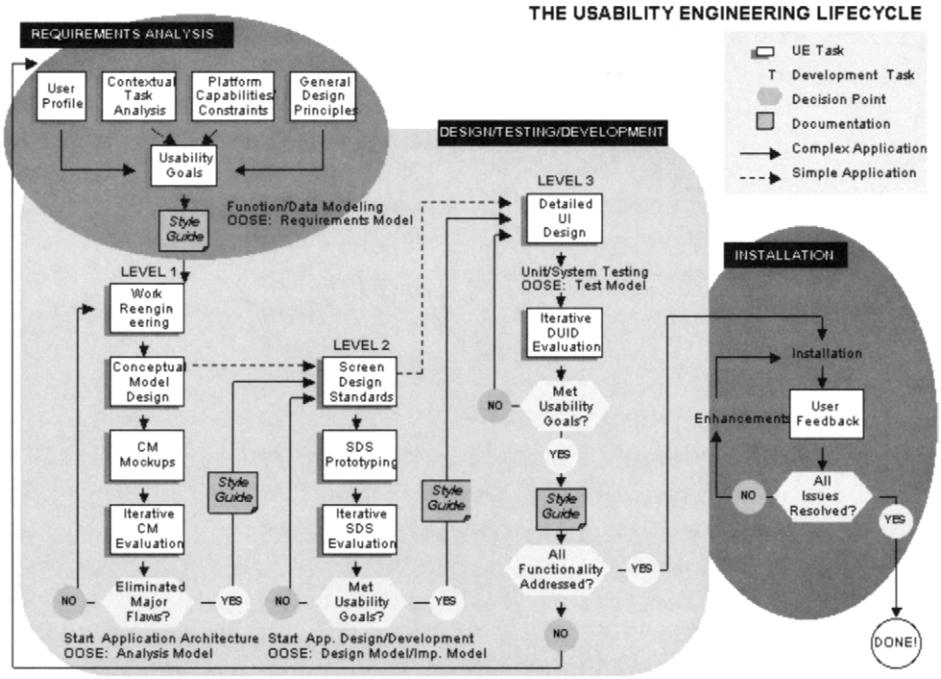

FIGURE 3.1 The Usability Engineering Lifecycle (taken from Mayhew, 1999; used with permission).

tasks. Each task has some conceptual goal that defines it. For example, the goal of the User Profile task is to gain a clear understanding of those characteristics of the intended user population that will have a direct bearing on which design alternatives will be most usable to them.

A *technique*, on the other hand, is a particular process or method for carrying out a task and for achieving a task goal. Usually there are a number of alternative techniques available for any given task. For example, for the User Profile task, alternative techniques include distributing user questionnaires and conducting user or user manager interviews. Generally, techniques vary in how costly and time consuming it is to execute them, in the quality and accuracy of the work products they generate, in how difficult they are for nonspecialists to learn and use, and in the sophistication of the technology required to carry them out.

The key to the general applicability and flexibility of the Usability Engineering Lifecycle lies in the choice of which *techniques* to apply to each task, *not* in the choice of which *tasks* to carry out. *All* the tasks identified in the Lifecycle

should be carried out for every development project involving interactive software in order to achieve required levels of usability. However, the approach to any given project can be adapted by a careful selection of *techniques* based on project constraints.

Each usability task in the overall Usability Engineering Lifecycle is briefly described in the following sections, with notes on adapting the tasks to Web development projects in particular.

3.2.1 Phase One: Requirements Analysis

User Profile. A description of the specific user characteristics relevant to user interface design (e.g., computer literacy, expected frequency of use, level of job experience) is obtained for the intended user population. This will drive tailored user interface design decisions, and also identify major user categories for study in the Contextual Task Analysis task discussed later.

The problems of doing a User Profile for a Web site or Web-enabled application are similar to those of doing one for a vendor company: Users are not readily accessible, and may not be known at all. However, developing a User Profile is still possible. The marketing department is often able to identify and get access to potential users. A shortcut for the User Profile task is interviewing marketing, sales, and sales support personnel or others who may have contact with actual current and potential users. And User Profile information can be solicited through the site itself after the Web site or Web-enabled application is implemented. This information can be used to update and improve the new versions of the Web site and to build new related Web sites and applications. A link can be embedded in the Web site that leads users to a User Profile questionnaire where incentives (e.g., discounts or raffle entries) are offered to the user for filling in the online survey (see Weiss, Chapter 19).

Contextual Task Analysis. A study of users' current tasks, workflow patterns, work environments and conceptual frameworks is made, resulting in a description of current tasks and workflow and an understanding and specification of underlying user goals in their identified environments. These will be used to set usability goals and drive Work Reengineering and user interface design.

The problems of doing a Contextual Task Analysis for a Web site or Web-enabled application are, again, very like those of doing one for a vendor company: the users are not easily accessible and may not be known at all, and the "work" may not currently be being performed by intended users. However, conducting a Contextual Task Analysis is still possible.

In a Contextual Task Analysis for a Web site or Web-enabled application, the focus might be more on what people *want and/or need*, rather than on how they currently *do* tasks. You can often get help from Marketing to identify and get access to potential users. You *can* do a Contextual Task Analysis of average people doing personal tasks at home, such as catalog ordering, planning travel, or buying a new car (Vaananen-Vainio-Mattila and Ruuska, 2000; Dray and Mrazek, 1996). You can also solicit task-related information from the Web site itself after the fact. You can have a feedback page and use feedback to update and maintain a site and to build new related Web sites and Web-enabled applications. You can also conduct some task analysis techniques, such as card sorting, via a Web site (see, for example, Weiss, Chapter 19).

Usability Goal Setting. Specific, *qualitative* goals reflecting usability requirements extracted from the User Profile and Contextual Task Analysis, and *quantitative* goals defining minimal acceptable user performance and satisfaction criteria based on a subset of high-priority qualitative goals, are developed. These usability goals focus later design efforts and form the basis for later iterative usability evaluation.

In most cases, at least when designing public Web sites or applications, *ease-of-learning* and *ease-of-remembering* goals will be more important than *ease-of-use* goals, because of the infrequency of use of the Web site. Most users do not visit a given Web site daily, and many often visit a site only once.

Ease of navigation and *maintaining context* will usually be very important *qualitative* goals for Web sites and applications.

Web designers need to be aware when formulating *quantitative performance* goals that system response time will limit and affect user performance, and that system response time will vary enormously depending on the users' platforms.

In many cases of Web site or Web-enabled application design, *relative quantitative* goals may be appropriate (e.g., "It must take no longer to make travel reservations on the Web site than it does with a travel agent by phone," or "It must take less time to make travel reservations on this site than on main competitors' sites").

Platform Capabilities/Constraints. The user interface capabilities and constraints (e.g., windowing, pull down menus, frames, animation) inherent in the technology platform chosen for the product (e.g., Microsoft Windows, Web browsers, or product-unique platforms) are determined and documented. These will define the scope of possibilities for user interface design.

Unlike some of the other Usability Engineering Lifecycle tasks, the Platform Capabilities and Constraints task will often be *more* complicated when designing a Web site or Web-enabled application than when designing traditional software applications. This is because usually (with the exception of the case of some

intranet applications) designers may have to assume a very large number and wide variety of hardware and software platforms.

Internet users' platforms will vary, possibly widely, in at least the following ways:

- Screen size and resolution
- Data transmission speed
- Activation of pop-up blockers, and security and spyware detection software
- Browser capabilities (varies by vendor and by version):
  - Controls available through the browser (vs. must be provided within the site or application)
  - Browser interpreters (e.g., version of HTML, Java)
  - Installed "helper applications" or "plug-ins" (e.g., multimedia players)

Web user interface designers need to design for the expected *range* of platform capabilities and constraints. For example, one common technique is to have a control at the entry point to a Web site or Web-enabled application that allows users to choose between a "graphics mode" and a "text mode." Thus, users with slow modems can "turn off" any graphics that would seriously degrade download time, and see an alternative text-only version of the site or application.

Similarly, if a Web site or Web-enabled application requires specific "helper applications" or "plug-ins," many are now designed to allow immediate downloading and installation of the required helper application or plug-in. The user interface to downloading and installing helper applications or plug-ins is still often not very user friendly, but at least providing the capability is a step in the right direction.

In general, while designers of intranets may be able to assume certain high-end platform parameters, designers of public Web sites need to be aware that if they take full advantage of all the latest Web capabilities, many users will find their Web site or Web-enabled application unusable. Care needs to be taken to provide alternative interfaces for users with lower-end platforms.

General Design Guidelines. Relevant general user interface design guidelines available in the usability engineering literature are gathered and reviewed. They will be applied during the design process to come, along with all other project-specific information gathered in the previous tasks.

Most general software user interface design principles and guidelines will be directly applicable to Web site and application design.

Things to bear in mind that do make designing for the Web a little different than designing traditional software include the following:

- Response times are slower and less predictable on the Web, limiting which design techniques are practical
- There are little or no existing comprehensive and widely accepted user interface *standards* for the Web (although various *guidelines* are available)
- Browsers and users, rather than designers and developers, may control much of the *appearance* of Web content
- Web users may be mainly discretionary and infrequent, increasing the need for "walk up and use" interfaces
- The World Wide Web is a huge and fluid space with fuzzy boundaries between sites. There is thus an increased need for navigational support and a "sense of place"

These differences are not quantitative, however. Rather, they are a matter of degree. Web platforms simply place more constraints on designers than do traditional platforms. Designing for the Web is somewhat like designing for traditional software 20 years ago—although the capabilities on the Web are now catching up fast.

3.2.2 Phase Two: Design/Testing/Development

Level 1 Design

Work Reengineering. Based on all requirements analysis data and the usability goals extracted from them, user tasks are redesigned at the level of organization and workflow to streamline work and exploit the capabilities of automation. No *visual* user interface design is involved in this task—just abstract organization of functionality and workflow design. This task is sometimes referred to as Information Architecture.

Sometimes you are actually simply engineering—rather than reengineering—work, because your Web site or Web-enabled application supports work unlike anything most of the intended users currently do (e.g., deciding on the structure for an information space users did not previously have access to). Nevertheless, you can still do a Contextual Task Analysis to discover users' needs and desires, and can base your initial work organization on this analysis.

In most cases, even when users do not currently do a particular job, they do already do something highly related to that job, and this can be the focus of a Contextual Task Analysis. In addition, once an initial release of a Web site or Web-enabled application is in production, you can perform another Contextual

Task Analysis to discover how it is being used and where it breaks down and use these insights to reengineer the underlying work models for later releases. And, just as when designing traditional software, you can still *validate* your reengineered work models empirically with evaluation techniques.

Conceptual Model Design. Based on all of the previous tasks, initial high-level design rules are generated for presenting and interacting with the application structure and navigational pathways (i.e., the information architecture). Screen design detail is *not* addressed at this design level.

The Conceptual Model Design is equally important in Web-site and Web-application design as in traditional software design. A Conceptual Model Design for a Web site might typically include rules that would cover the consistent presentation of:

- Site title and logo, including location
- Use of frames (e.g., for highest level links, context information, and page content)
- Links to different levels in the site map
- "You are Here" indicators on links
- Links versus other actions (e.g., "Submit" or "Search")
- Links versus non-links (e.g., illustrations)
- Inter versus intrasite links
- Inter versus intrapage links

On very simple Web site or Web-enabled application projects, it may not be necessary to formally document the Conceptual Model Design. Nevertheless, the Conceptual Model should be explicitly designed and validated.

Conceptual Model Mockups. Paper-and-pencil or prototype mockups (Snyder, 2003) of high-level design ideas generated in the previous task are prepared, representing ideas about how to present high-level functional organization and navigation. Detailed screen design and complete functional design are *not* in focus here.

Instead of paper foils or throw-away prototypes, in the case of Web sites and applications, the "mockups" could be partially coded products, for example, pages, frames, and navigational links with minimal page content detail.

Iterative Conceptual Model Evaluation. The mockups are evaluated and modified through iterative evaluation techniques such as formal usability testing, in which real, representative users attempt to perform real, representative tasks with minimal training and intervention, imagining that the mockups are a real

product user interface. This and the previous two tasks are conducted in iterative cycles until identified major usability "bugs" are engineered out of the Level 1 (i.e., Conceptual Model) design. Once a Conceptual Model is relatively stable, system architecture design can commence.

Remote usability testing is particularly well suited to testing Web sites and Web applications. It can replace (or complement) traditional usability testing in which the tester and user are side by side in the same location, and is more practical when users are widely dispersed geographically. The basic technique of remote usability testing involves giving a tester access to what is happening (or did happen) on the computer of a test user in another location. There are several ways to do this, including attended, real-time evaluations similar to traditional laboratory testing but conducted in real time over the Internet, instrumented methods that are unattended but otherwise similar to traditional testing techniques, and automated methods that unobtrusively collect usage data while a user is using a site (Perkins, 2002).

Level 2 Design

Screen Design Standards. A set of application- or site-specific standards and conventions for all aspects of detailed screen or page design is developed, based on any industry and/or corporate standards that have been mandated (e.g., Microsoft Windows or Apple Macintosh), the data generated in the Requirements Analysis phase, and the application- or site-unique Conceptual Model Design arrived at during Level 1 Design. Screen Design Standards will ensure coherence and consistency—the foundations of usability—across the user interface.

Screen Design Standards are just as important and useful in Web design as in traditional software design. Besides the usual advantages of standards, in a Web site they will help users maintain a *sense of place within a site,* because your site standards will probably be different from those on other sites.

On very simple Web site or Web-enabled application projects, it may not be necessary to formally document the Screen Design Standards. Nevertheless, the Screen Design Standards must be explicitly designed and validated.

Web design techniques (both good and bad) tend to be copied—perhaps other Web designers will copy your Screen Design Standards! Perhaps someday we will even have a set of universal Web Screen Design Standards supported by Web development tools, not unlike Microsoft Windows and Apple Macintosh standards. This would contribute greatly to the usability of the Web, just as the latter standards have done for traditional software.

Screen Design Standards Prototyping. The Screen Design Standards (as well as the Conceptual Model Design) are applied to design the detailed user

interface to selected subsets of product functionality. This design is implemented as a running prototype.

Instead of paper foils or throw-away prototypes, in the case of Web sites and applications, the prototypes can simply be partially coded products, for example, *selected* pages, frames and navigational links, now with complete page content detail.

Iterative Screen Design Standards Evaluation. An evaluation technique such as formal usability testing is carried out on the Screen Design Standards prototype, and then redesign and re-evaluate iterations are performed to refine and validate a robust set of Screen Design Standards. Iterations are continued until identified major usability bugs are eliminated and usability goals seem within reach.

Again, remote usability testing (Perkins, 2002) can be particularly useful when testing Web sites and applications.

Style Guide Development. At the end of the design and evaluate iterations in Design Levels 1 and 2, you have a validated and stabilized Conceptual Model Design and a validated and stabilized set of standards and conventions for all aspects of detailed screen design. These are captured in the document called the product Style Guide, which already documents the results of Requirements Analysis tasks. During Detailed User Interface Design, following the Conceptual Model Design and Screen Design Standards in the product Style Guide will ensure quality, coherence, and consistency, the foundations of usability.

For simple Web sites and applications, as long as good design processes and principles have been followed, documentation can be minimal and informal: a simple running list.

For complex Web sites and Web-enabled applications with many designers, developers and/or maintainers of a constantly evolving site or application, documenting Requirements Analysis work products and design standards is very important, just as it is on large, traditional software projects.

Level 3 Design

Detailed User Interface Design. Detailed design of the complete product user interface is carried out based on the refined and validated Conceptual Model Design and Screen Design Standards documented in the product Style Guide. This design then drives application or site development.

For simple Web sites or applications, designers might bypass documenting user interface design at the Conceptual Model Design and Screen Design Standards levels, and simply prepare Detailed User Interface Design specifications

directly from standards they have informally established at these earlier design levels. Developers can then code directly from these specifications.

For more complex Web sites or applications, the Conceptual Model Design and Screen Design Standards should usually be documented before this point. Then developers can code directly from an application or site Style Guide, or from Detailed User Interface Design specifications prepared based on a product Style Guide by the user interface designer.

Iterative Detailed User Interface Design Evaluation. A technique such as formal usability testing is continued during application or site development to expand evaluation to not-yet-assessed subsets of functionality and categories of users, and also to continue to refine the user interface and validate it against usability goals.

On projects developing relatively *simple* Web sites and applications, it might be more practical to combine the three levels of the design process into a single level, in which Conceptual Model Design, Screen Design Standards, and Detailed User Interface Design are *all* sketched out in sequence *before* any evaluation proceeds. Then a single process of design and evaluation iterations can be carried out.

In this case, Iterative Detailed User Interface Design Evaluation will be the first usability evaluation task conducted. Thus, evaluation must address all levels of design simultaneously. This is practical only if the whole site or application is fairly simple, which information-only (i.e., nontransactional) Web sites often are. It is important to remember that even if Detailed User Interface Design is drafted before any evaluation commences, it is still crucial to consider all the same design issues that arise in the Conceptual Model Design and Screen Design Standards tasks when conducting design in a three-level process.

In Web sites and applications of *intermediate complexity*, the first *two* design levels (Conceptual Model Design and Screen Design Standards) may be combined into a single process of design and evaluation iterations to validate them simultaneously, and then an additional process of design and evaluation can be carried out during the Detailed User Interface Design level. Alternatively, Level 1 can be carried out with iterative evaluation, and then Levels 2 and 3 can be collapsed into one iterative cycle. In either case, this will be the *second* usability evaluation cycle conducted, and can indeed focus mainly on Screen Design Standards and Detailed User Interface Design, because Conceptual Model Design will have been focused on during an earlier evaluation task.

Also, in the case of Web site or Web-enabled application design, one alternative is that mockups, prototypes, *and* application code can *all* simply be final code at different points of completion rather than paper foils or throw-away prototypes.

As in previous design levels, remote usability testing (Perkins, 2002) can be particularly useful when testing Web sites and applications.

3.2.3 Phase Three: Installation

User Feedback. After the product has been installed and in production for some time, feedback is gathered to feed into enhancement design, the design of new releases and/or the design of new but related products.

User feedback can be solicited directly from a Web site or Web-enabled application. This can be done by providing a link on the site taking users to a structured feedback page, or by offering direct e-mail from the site and asking users to provide free-form feedback. You can even have survey questions pop up, triggered by specific usage events. An advantage of the latter is that it collects feedback while the user's experience is fresh in his or her mind. The disadvantage, of course, is that users may find it irritating to be interrupted by a solicitation of this sort.

You might need to provide some incentive for users to take the time to provide feedback (see Weiss, Chapter 19), especially if you provide a lengthy, structured form (shorter forms probably work best). Possible incentives include entry in a raffle or discounts on products or services.

The user feedback techniques that lend themselves most easily to Web sites and Web-enabled applications include questionnaires (see Weiss, Chapter 19) and usage studies. Other techniques (e.g., interviews, focus groups and usability testing) are more difficult to employ since they require the identification and recruitment of users to meet in person with project team members, which may not be difficult on *Intranet* sites, but may be difficult on *Internet* sites.

To young Web designers or developers who launched their careers in the Internet age and have worked primarily on Web-development projects, the Usability Engineering Lifecycle may, at first glance, seem much too complex and time consuming to be practical in the fast-paced world of Web development. If you consider only the traditional and most reliable and thorough techniques for Lifecycle tasks, and typical timeframes for the development of very simple read-only Web sites, this is a fair assessment. For example, whereas I (Mayhew) have often conducted task analyses techniques that took several months to complete, and formal usability tests that took a month or more, I have also worked on Web development projects that from start to launch took a total of 8 to 12 weeks. Clearly you cannot spend several months conducting task analyses—just one of the first steps in the Usability Engineering Lifecycle—when the whole project must be completed in 2 to 3 months!

Two points must be made however. First, the Usability Engineering Lifecycle is highly flexible and adaptable through the selection of *techniques* applied to each task and the collapsing of Design Levels, as previously described, and can accommodate even projects with very limited timeframes. I (Mayhew) have in fact successfully adapted it to even 8-week Web-development projects.

Second, Web site functions were initially very simple compared to most traditional software applications, and so the fact that they typically took 8 to 12 weeks to develop, as compared to months or even years for traditional software applications, made some sense. Now, however, Web sites and applications have become more and more complex, and in many cases are much like traditional applications that happen to be implemented on a browser platform. The industry needs to adapt its notion of reasonable, feasible, and effective timeframes (and budgets) for developing complex Web-based applications, which simply are not the same as simple content-only Web sites. This includes adapting its notion of what kind of usability engineering techniques it should invest in.

In a report by Forrester Research, Inc. (Sonderegger, 2000), called "Scenario Design" (their term for usability engineering), it is pointed out that:

> **Executives Must Buy Into Realistic Development Time Lines and Budgets**
> The mad Internet rush of the late 1990s produced the slipshod experiences that we see today. As firms move forward, they must shed their misplaced fascination with first-mover advantage in favor of lasting strategies that lean on quality of experience.
>
> - Even single-channel initiatives will take eight to 12 months. The time required to conduct field research, interpret the gathered information, and formulate implementation specs for a new Web-based application will take four to six months. To prototype, build, and launch the effort will take another four to six months. This period will lengthen as the number of scenarios involved rises.
> - These projects will cost at least $1.5 million in outside help. Firms will turn to eCommerce integrators and user experience specialists for the hard-to-find-experts, technical expertise, and collaborative methodologies required to conduct Scenario Design. Hiring these outside resources can be costly, with run rates from $150 K to $200 K per month. This expenditure is in addition to the cost of internal resources, such as project owners responsible for the effort's overall success and IT resources handling integrations with legacy systems (p. 12).

We agree that 8 to 12 *months* is a more realistic timeframe (than 8 to 12 *weeks*) to develop a usable Web site or Web-enabled application that will provide a decent ROI. And, if this is the overall project timeframe, there is enough time to use traditional usability engineering techniques to more reliably ensure Web site usability, a major contributor to ROI. In my experience (Mayhew), depending on the complexity of a Web site or Web-enabled application, somewhere between $100,000 and $250,000 should pay for a reliable and thorough usability engineering program. This is a small fraction of the $1.5 million estimated by Forrester for all the outside help a Web site sponsor will need. And, as the rest of this chapter and other chapters in this volume illustrate, significant time and money invested in Web site or Web-enabled application usability will usually pay off.

3.3 GENERAL APPROACH

To cost-justify a usability engineering plan, you simply adapt a very generic and widely used cost-benefit analysis technique. Having laid out a detailed usability project plan based on Lifecycle tasks (see previous sections, and Mayhew, 1999), it is a fairly straightforward matter to calculate the costs of that plan. Then you need to calculate the benefits. This is a little trickier, and it is where the adaptation of the generic analysis comes into play (see later discussion). Then you simply compare costs to benefits to find out if and to what extent the benefits outweigh the costs. If they do to a satisfactory extent, then you have cost-justified the planned effort.

More specifically, first a usability engineering plan is laid out. The plan specifies particular techniques to employ for each Usability Engineering Lifecycle task, breaks the techniques down into steps, and specifies the personnel hours and equipment costs for each step. The cost of each task is then calculated by multiplying the total number of hours for each type of personnel by their effective hourly wage (fully loaded, i.e., including salary, benefits, office space, equipment, utilities, and other facilities), and adding up personnel costs across types. (Sometimes it is hard to get data on fully loaded wages for an organization. In this case, I (Mayhew) use a rule of thumb I have heard informally and simply double the before-tax annual salary, then divide by the typical number of hours a full-time worker is paid for in a year, usually about 2000. Even if my audience is unwilling to give me actual figures for fully loaded wages, they can contest—or not—my ballpark figure based on this rule of thumb.) Any equipment and other costs can be added in. Then the costs from all tasks are added to arrive at a total cost for the plan.

Next, the overall benefits of the specific usability engineering plan are predicted by selecting relevant benefit categories, calculating expected benefits by plugging project-specific parameters and assumptions into benefit formulas, and adding benefits across categories.

The list of possible benefits to consider is long, because usability engineering can lead to tangible benefits to all concerned, regardless of the type of organization or type of application. The development team realizes savings because problems are identified early, when they are cheap to fix. In vendor companies, the customer support team realizes a reduced customer support burden (although it may take time to show up, because a new interface may cause an initial surge in calls before it settles down). More usable e-commerce Web sites will have higher buy-to-look ratios, a lower rate of abandoned shopping carts, and increased return visits. Internal user productivity will be increased, and there will be lower user training costs.

The potential benefit categories selected in a particular cost-benefit analysis will depend on the type of organization taking on the development effort, including the usability engineering costs. In the case of a *development organization serving internal users*, benefits to the company as a whole might include:

- Increased user productivity
- Decreased user errors (and faster recovery when errors are made)
- Decreased training costs
- Savings gained from making changes earlier in the development lifecycle
- Decreased customer service calls from users

Benefits of usability engineering efforts to a *vendor company* might include:

- Increased sales
- Decreased customer service calls from users
- Savings gained from making changes earlier in the development lifecycle
- Reduced cost of providing training (if training is offered through the vendor company)

Note that although the primary benefit relevant to the development organization with internal users might be increased user productivity, this is not usually of direct concern to a vendor company (even though it should be). The vendor company is more concerned with selling more products and decreasing their customer support costs. Thus, in a cost-benefit analysis, you should focus atten-

tion on the potential benefits that are of most interest to the audience for the analysis.

Note that these benefits represent just a sample of those that might be relevant in these two types of organizations. Others might be included as appropriate, given the business goals of the organization and the primary concerns of the audience, and could be calculated in a similar fashion as shown in a later section.

In the case of Web sites and Web-enabled applications, the potential benefit categories relevant to a particular cost-benefit analysis will depend on the basic business model for the site. Benefit categories potentially relevant to different types of sites are summarized in Table 3.1.

Note that the relevant benefit categories for different types of Web sites and Web-enabled applications vary somewhat. In a cost-benefit analysis, you should focus attention on the potential benefits that are *of most relevance to the bottom line business goals for the site,* whether they are short term, long term, or both.

Again, note that these benefits represent just a sample of those that might be relevant in these types of sites, and do not address other possible benefits of usability in other types of sites. Others might be included as appropriate, given the business goals of the site sponsors and the primary concerns of the audience, and could be calculated in a similar fashion as those shown later in the chapter.

Finally, overall benefits are compared to overall costs to see if, and to what extent, the overall usability engineering plan is justified.

When usability practitioners are invited to participate in projects already in progress, which is often the case for external consultants, they have less chance of including all Usability Engineering Lifecycle tasks and of influencing overall schedules and budgets. They are more likely to have to work within already-committed-to schedules, platforms, and system architectures, to use shortcut techniques for Usability Engineering Lifecycle tasks, and to minimally impact budgets. Nevertheless, it is almost always possible to create a usability engineering plan that will make a significant contribution to a software development project, even when you come into the project relatively late. You can use the cost-benefit analysis technique to prepare and support even usability engineering plans that involve only parts of the overall Usability Engineering Lifecycle, and only shortcut techniques for tasks within it.

3.4 SAMPLE COST-BENEFIT ANALYSES

Let us consider a hypothetical usability engineering plan and see how its costs can be estimated. Then we will incorporate this plan into scenarios involving

Table 3.1 Potential Benefit Categories for Different Types of Web Sites

	Site Type				
Benefits	E-Commerce	Funded by Advertising	Product Information	Customer Service	Intranets
Increased buy-to-look ratios	✓				
Decreased abandoned shopping carts	✓				
Increased number of visits		✓			
Increased return visits	✓	✓			
Increased length of visits		✓			
Decreased failed searches	✓	✓			
Decreased costs of other sales channels	✓				
Decreased use of "Call Back" button (i.e., live customer service)	✓				✓
Savings resulting from making changes in earlier development lifecycle	✓	✓	✓	✓	✓
Increased "click through" on ads		✓			

Table 3.1 *Continued*

	Site Type				
Benefits	E-Commerce	Funded by Advertising	Product Information	Customer Service	Intranets
Increased sales leads			✓		
Decreased costs of traditional customer service channels				✓	
Decreased training costs					✓
Increased user productivity					✓
Decreased user errors					✓

development in four different types of projects, and see how you would conduct cost-benefit analyses of that plan for each project. The four scenarios involve the development of:

- An application for internal users
- A commercial application by a vendor company
- An e-commerce Web site
- A product information Web site

3.4.1 An Application for Internal Users

Imagine that a development organization is planning to develop an application for use by an *internal* user organization (e.g., within a bank or an insurance company). The project is of moderate complexity and cost, and will result in an application that will be used by 250 users. Once developed and installed, the

application is expected to be in production for approximately 5 years before any major revisions are made.

First, the final results of a cost-benefit analysis are presented. Then, in the steps that follow, the derivation of the final results are shown. Table 3.2 shows the overall calculation of the *cost* of a usability engineering plan proposed by the project usability engineer. The first column identifies the overall project phase. The second column identifies which Usability Engineering Lifecycle tasks (see Fig. 3.1) and techniques are planned in each phase. The third, fourth, fifth and sixth columns identify the number of work hours required by usability engineers, developers, managers, and users to complete each task. The last column summarizes the total cost of each task, based on the fully loaded hourly rates for each type of personnel. A total cost for the whole plan is given at the bottom of the table.

It is important to note that the usability engineering plan laid out in Table 3.2 is one specific to a particular project. Different plans could be devised, involving different phases, tasks and techniques, and the costs of these plans would vary accordingly. Also, for simplicity's sake, we have not included the costs of materials and equipment in this example. These could easily be estimated and added to the total cost of the usability engineering plan. Finally, note that the role "developers" is also a simplification. It might include not just engineers, but also graphic designers, business analysts, quality assurance (QA) staff, and so on, in fact any non–usability engineer and nonmanager who is expected to participate in the usability engineering tasks laid out in the plan could fall into this category.

In this hypothetical project, the project usability engineer has calculated the *predicted benefits* of carrying out this usability engineering plan *in the first year* of application installation, as shown in Table 3.3.

The predicted benefits *over the expected product lifetime* of the application (5 years) are also shown in Table 3.3. Comparing these benefits and costs, the project usability engineer argues that, as shown in Table 3.4, in the first year alone, a net benefit of $168,144.44 is predicted, and over the expected 5-year lifetime of the product, a net benefit of $1,158,422.22 is predicted. The project usability engineer expects the plan to be approved and funded based on this cost-benefit analysis.

Note that the simple analyses offered here do not consider the time value of money—that is, the money for the costs is spent at one point in time, and the benefits come later. Also, if the money was *not* spent on the costs, but instead was invested in some other way, it would likely increase in value. In our experience, the predicted benefits of usability engineering are usually so dramatic that these more sophisticated financial considerations aren't necessary to convince

Table 3.2 Cost of a Usability Engineering Plan

Phase	*Task (Technique)*	*Usability Engineer Hours at $175*	*Developer Hours at $175*	*Manager Hours at $200*	*User Hours at $25*	*Total Cost*
Requirements Analysis	User Profile (Questionnaire)	62	0	4	33	$12,475
	Contextual Task Analysis	138	8	8	60	$28,650
	Platform Capabilities and Constraints	16	6	0	0	$3,850
	Usability Goals	20	0	4	2	$4,350
Design/ Testing/ Development	Work Reengineering (Information Architecture)	80	0	0	16	$14,400
	Conceptual Model Design	80	8	0	8	$15,600
	Conceptual Model Mockups (Paper Prototype)	36	0	0	0	$6,300
	Iterative Conceptual Model Evaluation (Usability Test)	142	0	0	22	$25,400
	Screen Design Standards	80	8	0	8	$15,600
	Screen Design Standards Prototyping (Live Prototype)	28	80	0	0	$18,900
	Iterative Screen Design Standards Evaluation (Usability Test)	142	40	0	22	$32,400
	Detailed User Interface Design	80	8	0	8	$15,600
	Iterative Detailed User Interface Design Evaluation (Usability Test)	142	40	0	22	$32,400
	Totals	**1046**	**198**	**16**	**201**	**$225,925**

Table 3.3 Expected First Year and Lifetime Benefits for an Application for Internal Users

Benefit Category	*Benefit Value* First Year
Increased productivity	$199,652.78
Decreased errors	$47,916.67
Decreased training	$62,500.00
Decreased late design changes	$84,000.00
Total benefit	**$394,069.44**

Benefit Category	*Benefit Value* Lifetime (5 yrs)
Increased productivity 5 yrs	$998,263.89
Decreased errors × 5 yrs	$239,583.33
Decreased training × 1 yr	$62,500.00
Decreased late design changes × 1 yr	$84,000.00
Total benefit	**$1,384,347.22**

Table 3.4 Net Benefit Calculations for an Application for Internal Users

	Benefit	*Cost*	*Net Benefit*
First Year	$394,069.44	$225,925.00	$168,144.44
Lifetime (5 yrs)	$1,384,347.22	$225,925.00	$1,158,422.22

the audience of the analysis. However, if needed, these calculations based on the time value of money are presented in Karat (Chapter 4), and also in Bias *et al.* (2002).

In the following sections, we lay out step-by-step how the project usability engineer arrived at the final results stated previously.

1. Start with the Usability Engineering Plan

If it has not already been done, this is the first step in conducting a cost-benefit analysis. The usability engineering plan identifies which Usability Engineering Lifecycle tasks and techniques (see previous discussion and Mayhew, 1999) will be employed and breaks them down into required staff and hours. Costs can then be computed for these tasks in the next two steps.

The usability engineering plan for this sample analysis is shown in Table 3.2. It is important to note that there is not one correct usability engineering plan. This too is something that will vary across projects. The choice of technique for carrying out each task in the Usability Engineering Lifecycle will depend on project budgets, schedules, and complexity. Thus, the cost of the sample plan in the examples presented here should not be assumed—a project-unique plan must be designed around the parameters of a specific project, and then costs worked out as the example given here illustrates.

2. Establish Analysis Parameters

Most of the calculations for both planned costs and predicted benefits are based on project-specific parameters. These should be established and documented before proceeding with the analysis. Sample analysis parameters for our hypothetical project are given in Table 3.5.

It should be emphasized that when using the general cost-benefit analysis technique illustrated here, these particular parameter *values* should not be assumed. The particular parameter values of *your* project and organization should be substituted for those in Table 3.5. They will almost certainly be different from the parameters used in this example. For example, your application

Table 3.5 Analysis parameters for an application for internal users

Analysis Parameters	*Values*
Number of end users	250
User work days per year	230
User fully loaded hourly wage	$25
Developer fully loaded hourly wage	$175
Usability engineer fully loaded hourly wage	$175
Manager fully loaded hourly wage	$200
Ratio of early-to-late design changes	0.25
Expected system lifetime (yrs)	5
Current transactions per day	100
Current recovery time per error (2 min expressed as hrs)	0.033333333
Time per early design change (hrs)	8
Ratio of late-to-early design changes	4
Usability lab	In place

may be intended for many more (or fewer) than 250 users, and the fully loaded hourly wage (the costs of salary plus benefits, office space, equipment, utilities, and other facilities) of your personnel may be significantly lower or higher than those assumed in these sample analyses.

Note that, in general, certain parameters in a cost-benefit analysis have a major impact on the magnitude of potential benefits. For example, when considering *user productivity*—of primary interest to internal development organizations—the critical parameters are the *number of users*, the *volume of transactions*, and, to some extent also the *users' fully loaded hourly wage*. When there is a large number of users and/or a high volume of transactions, even very small performance advantages (and low hourly wages) in an optimized interface will add up quickly to significant overall benefits. On the other hand, where there is a small number of potential users, and/or a low volume of transactions, benefits may not add up to much even when the potential per-transaction performance advantage seems significant and the user hourly wage is higher.

For example, consider the following two scenarios. First, imagine a case in which there are 5000 users and 120 transactions per day per user. Even a half second advantage per transaction in this case adds up.

$$5000 \text{ users} \times 120 \text{ transactions} \times 230 \text{ days} \times 1/2 \text{ second} = 19{,}167 \text{ hours}$$

If the users' hourly rate is $25, the annual savings are:

$$19{,}167 \text{ hours} \times \$25 = \$479{,}175$$

This is a pretty dramatic benefit for a tiny improvement on a per transaction basis! On the other hand, if there were only 25 users, and they were infrequent users, with only 12 transactions per day, even if a per-transaction benefit of 1 minute could be realized, the overall benefit would be minor.

$$25 \text{ users} \times 12 \text{ transactions} \times 230 \text{ days} \times 1 \text{ minute} = 1{,}150 \text{ hours}$$

At $25 per hour, the overall annual productivity benefit will only be:

$$1{,}150 \text{ hours} \times \$25 = \$28{,}750$$

Thus, in the case of productivity benefits, costs associated with optimizing the user interface are more likely to pay off when there are many users and many transactions.

In the case of the *sales* benefit for a *vendor*, as in a later example, the critical parameter is usually *profit margin*. If the profit margin per product is low, then a very large number of additional sales would have to be achieved from usability alone for the usability costs to pay off. On the other hand, if the profit margin per product is high, then only a small number of increased sales from usability would be necessary to pay for the usability program. Similarly, in the case of an e-commerce or product information Web site, as in other examples given later, the critical parameters will be volume of visitors and profit margin per completed sales transaction. Thus, critical analysis parameters will directly determine how much can be invested in usability and still pay off.

3. Calculate the Cost of Each Usability Engineering Lifecycle Task in the Usability Engineering Plan

The cost of each individual task/technique listed in Table 3.2 was estimated by breaking the task/technique down into small steps, estimating the number of hours required for each step by different types of personnel, and multiplying these hours by the known fully loaded hourly wage of each type of personnel (if outside consultants or contractors are used, their simple hourly rate plus travel expenses would apply, and if external users are recruited to participate, they will be paid at some simple hourly rate or flat fee).

Fully loaded hourly wages are calculated by adding together the cost of salary, benefits, office space, equipment, and any other relevant overhead for a type of personnel, and dividing this by the number of hours paid for each year for that personnel type. The hourly rate used here for usability engineering staff is based on an informal average of typical current salaries of senior-level internal usability engineering staff and external consultants in my recent experience. (See www.upassoc.org/upa_publications/upa_voice/survey/2000_survey.html for a fairly recent salary survey of usability practitioners.) The hourly rate of developers was similarly estimated. (See, for example, www.payscale.com/salary-survey/vid-18644/fid-6886.) However, the fully loaded hourly rate figures used to generate this and the other sample cost-benefit analyses below are just examples, and you would have to substitute the actual hourly rates of personnel in your own organization in an actual analysis. Additional costs, such as equipment and supplies, could also be estimated and added into the total cost of each task/technique, although that was not done here for simplicity's sake.

Cost estimates for each usability engineering task/technique included in the usability engineering plan presented in Table 3.2 were calculated as presented in Tables 3.6 through 3.18. In these calculations, as shown in Table 3.6, usability engineers and developers are estimated to cost $175 per hour, managers are

Text continues on p. 72

Table 3.6 Cost of User Profile (Questionnaire)

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*
Conduct needs finding	4		2	2
Draft questionnaire	6			
Management feedback	2		2	2
Revise questionnaire	4			
Pilot questionnaire	4			4
Revise questionnaire	2			
Select user sample	4			
Distribute questionnaire/ get responses	8			25
Data analysis	8			
Data interpretation/ presentation	20			
Total hours	62	0	4	33
Times hourly rate	× $175	× $175	× $200	× $25
Equals	$10,850	+ $0	+ $800	+ $825 = **$12,475**

Table 3.7 Cost of Contextual Task Analysis

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*
Review requirements specs	12			
Interview project team/ user reps	16	8	6	2
Identify key actors/ use cases	6		2	2
In-context observations	40			40
Card sorting	32			16
Task Analysis documentation	32			
Total hours	138	8	8	60
Times hourly rate	× $175	× $175	× $200	× $25
Equals	$24,150	+ $1,400	+ $1,600	+ $1,500 = **$28,650**

Table 3.8 Cost of Platform Constraints and Capabilities

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Review documentation	4				
Interview developers	6	6			
Document constraints/ capabilities	6				
Total hours	16	6	0	0	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$2,800	+ $1,050	+ $0	+ $0	= **$3,850**

Table 3.9 Cost of Usability Goals

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Draw from User Profile	4				
Draw from Contextual Task Analysis	4				
Research business goals	4		2	2	
Formulate/prioritize/ document goals	8		2		
Total hours	20	0	4	2	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$3,500	+ $0	+ $800	+ $50	= **$4,350**

Table 3.10 Cost of Work Reengineering (Information Architecture)

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Review all Requirements Data and Usability Goals	8				
Design Draft Information Architecture	24				
Validate Draft Information Architecture (Reverse Card Sorting)	24			16	
Document Draft Information Architecture	24				
Total hours	80	0	0	16	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$14,000	+ $0	+ $0	+ $400	= **$14,400**

Table 3.11 Cost of Conceptual Model Design

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Review all Requirements Data, Usability Goals and Information Architecture	8				
Design Draft Conceptual Model	32	8		8	
Document Draft Conceptual Model	40				
Total hours	80	8	0	8	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$14,000	+ $1,400	+ $0	+ $200	= **$15,600**

Table 3.12 Cost of Conceptual Model Mockup (Paper Prototype)

Step	*Usability Engineer's Hours*		*Developer's Hours*		*Manager's Hours*		*User's Hours*		
Select functionality	4								
Create paper prototype foils	32								
Total Hours	36		0		0		0		
Times Hourly Rate	× $175	×	$175	×	$200	×	$25		
Equals	$6,300	+	$0	+	$0	+	$0	=	**$6,300**

Table 3.13 Cost of Iterative Conceptual Model Evaluation (Usability Test)

Step	*Usability Engineer Hours*		*Developer Hours*		*Manager Hours*		*User Hours*		
Design/develop test materials	32								
Design/assemble test environment	4								
Pilot test/revise materials	10						6		
Run test/collect data (2 usability engineers)	32						16		
Collate data	16								
Analyze/interpret data, formulate Redesign	24								
Document/present conclusions	24								
Total hours	142		0		0		22		
Times hourly rate	× $175	×	$175	×	$200	×	$25		
Equals	$24,850	+	$0	+	$0	+	$550	=	**$25,400**

Table 3.14 Cost of Screen Design Standards

Step	*Usability Engineer Hours*		*Developer Hours*		*Manager Hours*		*User Hours*		
Review all requirements Data, Usability Goals, Information Architecture and Conceptual Model Design	8								
Design Draft Screen Design Standards	32				8		8		
Document Draft Screen Design Standards	40								
Total hours	80		8		0		8		
Times hourly rate	× $175	×	$175	×	$200	×	$25		
Equals	$14,000	+	$1,400	+	$0	+	$200	=	**$15,600**

Table 3.15 Cost of Screen Design Standards Prototyping (Live Prototype)

Step	*Usability Engineer Hours*		*Developer Hours*		*Manager Hours*		*User Hours*		
Select functionality	4								
Prepare design specification	24								
Build live prototype			80						
Total hours	28		80		0		0		
Times hourly rate	× $175	×	$175	×	$200	×	$25		
Equals	$4,900	+	$14,000	+	$0	+	$0	=	**$18,900**

Table 3.16 Cost of Iterative Screen Design Standards Evaluation (Usability Test)

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Design/develop test materials	32				
Design/assemble test environment	4	32			
Pilot test/revise materials	10	8		6	
Run test/collect data (2 usability engineers)	32			16	
Collate data	16				
Analyze/interpret data, formulate Redesign	24				
Document/present conclusions	24				
Total hours	142	40	0	22	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$24,850	+ $7,000	+ $0	+ $550	= **$32,400**

Table 3.17 Cost of Detailed User Interface Design

Step	*Usability Engineer Hours*	*Developer Hours*	*Manager Hours*	*User Hours*	
Review all Requirements Data, Usability Goals, Information Architecture, Conceptual Mode Design and Screen Design Standards	8				
Design Draft Detailed User Interface Design	32	8		8	
Document Draft Detailed User Interface Design	40				
Total hours	80	8	0	8	
Times hourly rate	× $175	× $175	× $200	× $25	
Equals	$14,000	+ $1,400	+ $0	+ $200	= **$15,600**

Table 3.18 Cost of Iterative Detailed User Interface Design Evaluation (Usability Test)

Step	*Usability Engineer Hours*		*Developer Hours*		*Manager Hours*		*User Hours*		
Design/Develop Test Materials	32								
Design/Assemble Test Environment	4		32						
Pilot test/revise materials	10		8				6		
Run test/collect data (2 usability engineers)	32						16		
Collate data	16								
Analyze/interpret data, formulate redesign	24								
Document/present conclusions	24								
Total hours	142		40		0		22		
Times hourly rate	× $175	×	$175	×	$200	×	$25		
Equals	$24,850	+	$7,000	+	$0	+	$550	=	**$32,400**

estimated to cost $200 per hour and users to cost $25 per hour. The total cost of each task/technique shown in Tables 3.6 through 3.18 are used in Table 3.2 to calculate the total cost of the whole usability engineering plan.

4. Select Relevant Benefit Categories

As shown in Table 3.3, the project usability engineer selected four benefit categories relevant to this application for internal users to include in the cost-benefit analysis:

1. Increased productivity
2. Decreased errors
3. Decreased training
4. Decreased late design changes

These seemed to be of most relevance to building an application for an internal user population. Other benefits might have been included, for example, decreased cost of user support time, but just these four were selected to keep the analysis simple and conservative. As already discussed, the best benefit cate-

gories to include in a cost-benefit analysis will depend on the type of project and the intended audience for the analysis.

In this case the project usability engineer expects to achieve increased productivity by focusing on streamlining across-screen navigation within tasks, by minimizing typing and mouse clicks on individual screens, and by designing to facilitate scanning and interpreting displays. He or she expects to decrease errors both by following well-established design principles during design and by detecting and eliminating common errors through usability testing. He or she expects to decrease training time by designing a consistent, rule-based user interface architecture which matches users' knowledge and expectations and in which the smallest number of rules accounts for the widest scope of functionality. Finally, late design changes will be minimized and replaced by less expensive early design changes by following an iterative design process which incorporates usability inspection and testing.

Table 3.3 summarizes the predicted magnitude of each of these benefits and adds them to predict a total benefit. What follows is an explanation of how benefit predictions in each category were derived.

5. *Predict Benefits*

Benefits are predicted in each selected benefit category by doing some simple arithmetic based on project-specific analysis parameters and some simple assumptions. The project parameters in this case are laid out in Table 3.5. The benefits assumptions are given in Table 3.19.

In the case of productivity, the relevant *parameters* are (from Table 3.5):

- The total number of users
- The number of days each user works per year

Table 3.19 Benefits Assumptions for an Application for Internal Users

Benefit Assumptions			
Increased Productivity	Decreased Errors	Decreased Training	Decreased Late Design Changes
Decr. time/ transaction (5 sec = 0.001389 hrs)	1 error eliminated/ day	10 hrs saved off current 1 wk training	20 changes made early

- The number of transactions each user currently performs each working day
- The users' fully loaded hourly wage

The *assumption* made regarding increased productivity (from Table 3.19) is that:

- Each transaction will take 5 seconds (0.001389 hours) less on a user interface developed with the usability engineering plan than on a user interface developed without the usability engineering plan

This single assumption is the crux of the whole cost-benefit analysis. While *costs* can be calculated with a high degree of confidence based on past experience, and all the *parameters* fed into the analysis are known facts, the *assumptions* made are just that—assumptions—rather than known facts or guaranteed outcomes. The audience for the analysis is asked to accept that these assumptions are reasonable, and they must to be convinced by the overall analysis.

Note that *any* cost-benefit analysis for *any* purpose must ultimately include some assumptions that are really only predictions of the likely outcome of investments of various sorts. The whole point of a cost-benefit analysis is to try to evaluate in advance, in a situation in which there is some element of uncertainty, the likelihood that an investment will pay off. The trick is basing the predictions of uncertainties on a firm foundation of known facts. In the case of a cost-benefit analysis of usability engineering, there are several foundations upon which to formulate sound assumptions regarding benefits.

First, there is 25 years of published research that shows measurable and significant performance advantages of specific user interface design alternatives under certain circumstances. Examples of design alternatives for which performance data exist include the following:

- Use of color
- Choice of input devices
- Use of windowing
- Use of direct manipulation
- Screen design details
- Menu structure

Benefit assumptions can thus be defended by referring to studies of such design alternatives. Available studies that explore the relative benefits of different design alternatives typically vary one narrow aspect of design, such as fill-in form design, use of windows, use of color, or system response time, keeping all other

design variables constant, and measure human performance on some simple, well-defined tasks.

From these studies, we can extrapolate to make some reasonable predictions about the order of magnitude of differences we might expect to see in user interfaces that have been optimized through the execution of a usability engineering plan. The research does not provide simple, generic answers to design questions. However, what the research does provide are *general ideas* of the *magnitude of performance differences that can occur between optimal and suboptimal interface design alternatives.* The basic benefit assumptions made in any cost-benefit analysis can thus be generated and defended in part by simply referring to the wide body of published research data that exists.

Besides citing relevant general research literature, there are other ways to arrive at and defend one's benefit assumptions in a cost-benefit analysis. Actual case histories of the benefits achieved as a result of applying usability engineering techniques are very useful in helping to defend the benefits assumptions of a particular cost-benefit analysis. A few published case histories exist (e.g., Karat, 1989); Wixon and Wilson (1997) and Whiteside *et al.* (1988) reported that across their experience with many projects over many years, they found that they averaged an overall performance improvement of about 30% when at least 70 to 80% of the problems they identified during testing were addressed by designers. Landauer (1996, pp. 221–223) also cites an impact of usability engineering between zero and several hundred percent (with an average of 50 percent) on a set of 18 projects reported in the literature, depending on the complexity and type of product. Also, in Chapter 16, I (Mayhew) describe a case study based on a real project, in which task times were improved on average by 21% on a redesigned user interface as compared to an original user interface. Across eight tasks, which took an average of 2 minutes on the original interface, this translated into an average time savings of 26 seconds per task. This is just another example of a documented case in which an alternative interface to a specific task significantly increased productivity.

But even anecdotes are useful. For example, a colleague working at a vendor company once told me (Mayhew) that she had compared customer support calls on a product for which they had recently developed and introduced a new, usability-engineered release. Calls to customer support *after* the new release were decreased by 30%. This savings greatly outweighed the cost of the usability engineering effort.

Nielsen (1993, p. 84) informally reports a case involving the interface to the installation process for an upgrade of a spreadsheet package. When the upgrade was shipped, customers needed an *average of two 20 minute calls each* to customer support to correctly install the upgrade. Support calls to the vendor cost them

an average of $20 per 5 minutes to service—thus, the support cost *per customer* for this product was about $160. Unfortunately, the profit margin on the upgrade product was only $70 per customer—thus, not only was the ROI on the product eroded but the upgrade product actually *cost* the vendor nearly $100 per customer! The cost of the support costs was all a result of usability problems that probably could have been detected and fixed fairly cheaply prior to releasing the upgrade product.

In fact, some e-commerce Web sites have failed and been shut down in large part because of poor usability. (Souza, 2000, cites boo.com and levi.com as examples.) All these anecdotes can serve to strengthen specific cost-benefit analyses that make conservative assumptions regarding benefits.

In addition, experienced usability engineers can draw upon their own general experience evaluating and testing software user interfaces and their specific experiences with a particular development organization to defend benefit assumptions that they have incorporated into cost-benefit analyses. Familiarity with typical interface designs from a development organization allows the usability engineer to decide how much improvement to expect from applying usability engineering techniques in that organization. If the designers are generally untrained and inexperienced in interface design and typically design poor interfaces, the usability engineer would feel comfortable and justified defending more aggressive benefits assumptions. On the other hand, if the usability engineer knows the development organization to be quite experienced and effective in interface design, then more conservative predictions of benefits would be appropriate, on the assumption that usability engineering techniques will result in fine tuning of the interface but not radical improvements. The usability engineer can assess typical interfaces from a given development organization against well-known and accepted design principles, against past usability test results, and against the research literature to help defend specific assumptions made when estimating benefits.

In general, it is usually wise to make *very conservative* benefit assumptions for several reasons. First, any cost-benefit analysis has an intended audience, who must be convinced that benefits will most likely outweigh costs. Assumptions that are very conservative are less likely to be challenged by the relevant audience, thus increasing the likelihood of acceptance of the analysis conclusions. In addition, conservative benefits assumptions help to manage expectations. It is always better to achieve a greater benefit than was predicted in the cost-benefit analysis, than to achieve less benefit, even if the benefits still outweigh the costs. Having underestimated benefits will likely make future cost-benefit analyses more credible and more readily accepted. Also, it is important to realize that some validly predicted benefits may be canceled out by other non–usability-

related changes, such as decreases in user morale and motivation, decreased system reliability, an economic downturn, new competition in the marketplace and so on (see Wilson and Rosenbaum, Chapter 8). Having made conservative benefits predictions decreases the possibility that other factors will completely wipe out any benefits that result from improved usability.

Returning to the explanation of the derivation of benefit predictions, we look at the analysis parameters used in Table 3.5, and the benefit assumptions for each benefit category given in Table 3.19. The project usability engineer selected these assumptions believing they are very conservative. In presenting the analysis, he or she cites some of the literature mentioned previously in this chapter that shows up to 20 to 30% savings in task time on one interface relative to another, and points out that this analysis assumes only a modest 4% increase in productivity. He or she also points out that most current internal applications take 1 week to train because there is a lack of consistency in the user interfaces of those applications, and users must memorize many cryptic codes and unclear error messages. Knowing this allows the usability engineer to make the case that an interface in which a small number of rules explains a wide scope of functionality, and in which the user needs to memorize less will be teachable in a significantly shorter period of time. The assumption that the new interface will eliminate one error per user per day is extremely conservative, and the usability engineer can cite internal statistics showing high typical user error rates on current internal applications. Finally, to defend the assumption about the relative cost of early versus late design changes, the project usability engineer cites a classic paper in the literature (Mantei and Teorey, 1988).

Table 3.20 shows the calculation of the total predicted benefit in each benefit category, based on parameters and assumptions. In the case of increased productivity, multiplying the number of users by days per user, transactions per day, hours saved per transaction, and hourly rate results in the total benefit given in Table 3.3 for this benefit category: $199,652.78. Benefit assumptions for the other three benefit categories can be seen in Table 3.19, calculations of total benefit predictions in each of these three other categories can be seen in Table 3.20, and these total benefits per category are added together in Table 3.3 to arrive at a total benefit in the first year alone, and a lifetime benefit over an assumed 5-year application lifetime.

6. Compare Costs to Benefits

Having calculated the costs of a particular usability engineering plan and the total benefits predicted to result from executing that plan as compared to not executing it, the next step is simply to subtract the total costs from the total

Table 3.20 Benefits Calculations for an Application for Internal Users

Increased Productivity											
No. Users		No. Days		No. Transactions		Hrs Saved/ Transaction		Hourly Rate		Total	
250	×	230	×	100	×	0.001389	×	$25	=	**$199,652.78**	

Decreased Errors											
No. Users		No. Days		No. Eliminated Errors		Hrs Saved per Error		Hourly Rate		Total	
250	×	230	×	1.0	×	0.033333	×	$25	=	**$47,916.67**	

Decreased Training						
No. Users		Hours Saved per User		Hourly Rate		Total
250	×	10	×	$25	=	**$62,500.00**

Decreased Late Design Changes						
Cost of Early Changes						
No. Changes		Hours per Change		Hourly Rate		Total
20	×	8	×	$175	=	**$28,000.00**

Cost of Late Changes				
Cost of Early Changes		Ratio of Late to Early Changes		Total
$28,000.00	×	4	=	**$112,000.00**

Savings of Early Changes Relative to Late Changes				
Cost of Late Changes		Cost of Early Changes		Total
$112,000.00	–	$28,000.00	=	**$84,000.00**

benefits to arrive at a net benefit. In this example, this calculation is shown in Table 3.4. The analysis predicts a clear net benefit ($168,144.44) in the first year alone, and a dramatic net benefit ($1,158,422.22) over the expected application lifetime.

Our project usability engineer's initial usability engineering plan appears to be well justified. It is a fairly aggressive plan in that it includes all Lifecycle tasks, and the most reliable and thorough techniques for each task. Given the very clear net benefit, the usability engineer would be wise to stick with this aggressive plan and submit it to project management for approval and funding.

If the net benefit had been marginal, or if there had been a net cost, then the usability engineer would be well-advised to go back and rethink the plan, scaling back to shortcut techniques for some tasks. Perhaps, for example, the usability engineer should plan to do only a shortcut User Profile by interviewing user management, a shortcut Task Analysis consisting of just a few rounds of contextual observations and interviews with users, and then do just one iterative cycle of usability testing on a complete, detailed design, to catch major flaws and be sure the predicted benefits have been achieved. Of course, this would make the predictions more risky, and suggest an even more conservative analysis.

As explained earlier, to plan the budget for a usability engineering program, it makes sense to start out by calculating the costs of the most aggressive usability engineering program that you would like to implement, including the more reliable and thorough techniques for most, if not all, Lifecycle tasks. If predicted benefits outweigh costs dramatically, as they usually will when critical parameters are favorable, then you can easily make a good argument for even the most aggressive usability engineering program, because only the most conservative claims concerning potential benefits have been made, and therefore can be defended easily.

If, however, benefits and costs in the initial calculation seem to match up fairly closely, then you might want to consider scaling back the planned usability engineering program, maybe even to just a bare-bones plan, with more shortcut techniques applied for each Lifecycle task.

To illustrate this planning strategy, consider the following two scenarios. First, revisit the example above, which involved building a system for 250 internal users. Fairly conservative assumptions were made concerning benefits: task time reduced by 5 seconds, training time reduced by 10 hours, one error eliminated per day per user at 2 minutes saved per error. Even with these conservative assumptions, the aggressive usability engineering plan was predicted to payoff in the first year, with net benefits continuing to accrue dramatically after that.

In fact, if you had made the more aggressive, yet still realistic, benefits assumptions of training time reduced by 20 hours (rather than by 10), two errors eliminated per user per day (rather than just 1), and task time reduced by 15 seconds (rather than just by 5 seconds), the benefits would have added up to $903,839.58 in the first year alone, outweighing the costs of $225,925 by $677,914.58, and to $3,683,197.92 over 5 years, outweighing the costs by $3,457,272.90. Thus, you could argue, even the most conservative assumptions predict a fairly dramatic payoff of a comprehensive usability engineering program, but the likelihood is that the payoff will be higher still.

In contrast, suppose you again started out by estimating a comprehensive usability engineering program to cost $225,925. In this case, however, suppose that there are only 50 intended users (instead of 250) performing 50 transactions per user per day (instead of 100). In this case, calculations using the original, more conservative, benefits assumptions would show a loss until well into the second year, and a 5-year lifetime net benefit of only $18,318.06.

Even though the benefits assumptions were conservative, and although a first year loss is not necessarily a bad thing, it still seems risky to make an aggressive investment that, based on conservative assumptions, really doesn't show a significant payoff even over the course of 5 years. In this case, you would want to scale back the planned usability engineering program and its associated costs. Because the benefits assumptions made were so conservative, it is likely that they will be achieved even with a minimal usability effort. Thus, the cost-benefit analysis technique can be used to "what if" in order to plan a level of usability engineering effort that is most likely to pay for itself.

3.4.2 A Commercial Application by a Vendor Company

In this section, we turn to another hypothetical example—this time, cost justification of a usability engineering plan in the case of a vendor company selling a software package. The analysis process will be very similar—it is primarily the benefit categories that will be different. In the case of a software vendor company, the primary benefit of interest is not user productivity or user errors, but increased sales and decreased customer support costs.

We will assume exactly the same usability engineering plan laid out in the previous example and presented in Table 3.2. Table 3.21 provides the summary of predicted benefits, and Table 3.22 provides the net benefit calculations. Here it can be seen that a net benefit of $361,408.33 is expected in the first year, and a net benefit of $728,075.00 is expected over a system lifetime of 3 years.

In the following sections we lay out step by step how the project usability engineer arrived at these conclusions.

Table 3.21 Expected First Year and Lifetime Benefits for a Commercial Application by a Vendor Company

Benefit Category	*Benefit Value* First Year
Increased sales	$100,000.00
Decreased customer service calls from users	$183,333.33
Decreased training	$220,000.00
Decreased late design changes	$84,000.00
Total benefit	**$587,333.33**

Benefit Category	*Benefit Value* Lifetime (3 Yrs)
Increased Sales × 1	$100,000.00
Decreased customer service calls from users × 3 yrs	$550,000.00
Decreased Training × 1 yr	$220,000.00
Decreased Late Design Changes × 1 yr	$84,000.00
Total benefit	**$954,000.00**

Table 3.22 Net Benefit for a Commercial Application by a Vendor Company

	Benefit	*Cost*	*Net Benefit*
First year	$587,333.33	$225,925.00	$361,408.33
Lifetime (3 yrs)	$954,000.00	$225,925.00	$728,075.00

1. Start with the Usability Engineering Plan

We will assume the same usability engineering plan as in the previous example, shown in Table 3.2.

2. Establish Analysis Parameters

Analysis parameters for this sample analysis are given in Table 3.23. Here it can be seen that while some parameters are the same as in the previous example

Table 3.23 Analysis Parameters for a Commercial Application by a Vendor Company

Analysis Parameters	*Values*
Current typical sales (in units)	10,000
User fully loaded hourly wage:	$25
Developer fully loaded hourly wage:	$175
Usability engineer fully loaded hourly wage:	$175
Manager fully loaded hourly wage:	$200
Customer support fully loaded hourly wage:	$50
Trainer fully loaded hourly wage:	$50
Max no. users per training class	20
Expected system lifetime (yrs):	3
Profit margin per unit	$100
Average length of customer support call (10 min expressed as hrs)	0.166667
Time per early design change (hrs):	8
Ratio of late to early design changes:	4
Usability lab:	In place

(e.g., personnel hourly wages), some are different (e.g., current typical annual sales and profit margin per unit), in part because of the different benefit categories that will be included in this analysis.

3. Calculate the Cost of Each Usability Engineering Lifecycle Task in the Usability Engineering Plan

Because all analysis parameters relative to computing costs are assumed to be the same as in the previous example, the cost calculations for each task in the plan, and thus the total cost, are the same, as shown in Table 3.2.

Note that this time the user rate of $25 per hour is not based on a typical user's fully loaded hourly wage, which it was in the previous analysis of traditional software development for internal users. Instead, the user rate is based on the assumption that test users in the case of a commercial software package will have to be recruited from the general public or from customer organizations to participate in usability engineering tasks/techniques, and that they will be paid at a rate of $25 an hour for their time.

4. Select Relevant Benefit Categories

In this example, the benefit categories differ somewhat from those in the previous example. They include the following:

- Increased sales
- Decreased customer service calls from users
- Decreased training costs
- Decreased late design changes

A wide variety of trade magazines and product review companies purchase and/or receive for free the new software releases that are sent to market. Approximately 15% (Nielsen, 1993) of reviews in microcomputer trade journals were devoted to analyzing the *user friendliness* or *usability* of new software products a decade ago—undoubtedly more are now. Many newspapers, such as the *New York Times*, the *Financial Times*, and the *Wall Street Journal* have weekly columns that evaluate software, and even general discussion of functionality in such columns often make comments on ease-of-use. Potential customers read these columns and make purchase decisions based on the evaluations, including the user interface portion. *Consumer Reports* has been including evaluations of the user interfaces to consumer products, as well as investigating their reliability and safety. Trade journals for products in the industry perform similar services for the customer.

In addition to publications, software products often undergo reviews from user groups and companies whose sole purpose is to review major expensive software products. The trend has been to include a review of the user interface as well as the functionality of the software. Later adopters call companies who have installed the product earlier for reviews.

While it is difficult to predict exactly what impact usability engineering efforts will have on product sales, it should be clear that usability is now an aspect of competitive edge. Making a conservative prediction of increased sales in your cost-benefit analysis is likely to be accepted by the relevant audience, especially if they are reminded of the facts just described.

To predict increased sales benefits, we can consider relevant market forces, such as current market share, trends of the market, and strengths and weaknesses of the competition, and then choose a conservative and realistic assumption concerning the number of new sales that could potentially be attributed to increased usability alone. Then we simply multiply this number by the known profit margin of the product.

Another significant benefit category for vendor companies is decreased customer support costs. Today's customer service operation is an extensive business. One major software vendor has so many callers on its customer hotline that the company has hired its own disk jockey to play music and give software advertisements while customers are on hold. An interface that is understandable and easy to learn will generate fewer customer requests for help and thus lower the number of customer support staff needed to handle the customer hotline. This reduction in personnel costs can be estimated as a potential benefit of applying usability engineering techniques during development.

In addition, we will assume in this example that the vendor provides training. Whether the vendor bundles the cost of training in with the application or sells the training as a separate product, it is crucial to keep the cost down to compete in the marketplace. Finally, vendors are just as interested in keeping the cost of late design changes down as any other kind of development organization, so we will include that benefit in our analysis as well.

The project usability engineer expects to increase sales by accomplishing a user interface design that has usability features that are easily demonstrable during the sales process and that will get good reviews for usability in relevant trade journals. He or she expects to decrease customer support calls by designing an interface that is more rule-based, consistent, and predictable, and that also builds off users' current knowledge and skills more effectively. Arguments regarding how he or she can decrease the costs of training and late design changes would be the same as those given in the previous example of an application for internal use.

5. *Predict Benefits*

Table 3.24 shows the basic assumptions made regarding the magnitude of the benefit in each chosen benefit category. Table 3.25 shows the benefit calculations

Table 3.24 Benefits Assumptions for a Commercial Application by a Vendor Company

Benefit Assumptions			
Increased Sales	Decreased Customer Service Calls	Decreased Training	Decreased Late Design Changes
Incr. sales by 10%	Eliminated 2 calls per customer per yr	8 hrs saved off current 2 day training	20 changes made early

Table 3.25 Benefits Calculations for a Commercial Application by a Vendor Company

Increased Sales						
Current Sales		Rate of Increase		Profit margin per unit		Total
10,000	×	0.10	×	$100.00	=	**$100,000.00**

Decreased customer service calls from users								
No. Customers		No. Calls eliminated		No. Hours saved per call		Customer support hourly rate		Total
11,000	×	2	×	0.166667	×	$50	=	**$183,333.33**

Decreased Training								
No. Customers		Max No. Customers/Class		No. Hrs Saved/Class		Trainer Hrly Rate		Total
11,000	÷	20	×	8	×	$50	=	**$220,000.00**

Decreased Late Design Changes						
Cost of Early Changes:						
No. Changes		Hrs/Change		Developer Hrly rate		Total
20	×	8	×	$175	=	**$28,000.00**

Cost of Late Changes:				
Cost of Early Changes		Ratio of Late to Early Changes		Total
$28,000.00	×	4	=	**$112,000.00**

Savings of Early Changes Relative to Late Changes:				
Cost of Late Changes		Cost of Early Changes		Total
$112,000.00	–	$28,000.00	=	**$84,000.00**

for each benefit category, arrived at by incorporating both analysis parameters and benefit assumptions.

The project usability engineer based the conservative assumption regarding increased sales on statistics from the marketing department regarding the current rate of lost sales opportunities attributed to usability issues. He or she based the very conservative assumption regarding decreased customer support costs on statistics from the customer support organization showing a very high rate of calls on existing products attributable to usability issues. He or she knows the typical training class on existing products is currently two days, and argues a more rule-based interface with less required rote memorization can be taught in half the time.

6. Compare Costs to Benefits

Again, Table 3.22 shows the net benefit calculations based on comparing the total benefits calculated in Table 3.21 with the total costs calculated in Table 3.2. The fairly aggressive usability engineering plan seems more than justified, even with relatively conservative benefits assumptions. An intangible benefit not explicitly factored into this analysis is the additional customer loyalty that will undoubtedly be built by providing a more user-friendly product. The usability engineer can point this out in the presentation of the analysis to the stakeholder audience.

3.4.3 An E-Commerce Site

The previous two examples illustrate a cost-benefit analysis conducted for a usability engineering plan in the context of software development projects based on traditional platforms such as Microsoft Windows. The basic analysis process is really no different for any kind of application on any sort of platform—the main difference is in the choice of benefit categories. We now present two examples of justifying a usability engineering effort on Web development projects.

In this example, imagine a Web development organization is planning to redesign an existing e-commerce site that is not producing the ROI hoped for. Traffic statistics are available from the existing site. As before, we first present the final results of a cost-benefit analysis of including a usability engineering effort in the redesign project. Then, in the steps that follow, the derivation of the final results are shown.

Table 3.26 Expected Monthly Benefits for an E-Commerce Web Site

Benefit Category	*Benefit Value per Month*
Increased buy-to-look ratio	$12,500.00
Decreased abandoned shopping carts	$12,500.00
Decreased usage of "Call Back" button	$3,125.00
Total monthly benefit	**$28,125.00**

Table 3.27 Net Benefit Calculations for an E-Commerce Web Site

Benefits/Month	*Total Cost*	*Payoff Period in Months (Cost ÷ Benefit)*	*Net Benefit* (First Year)
$28,125.00	$225,925.00	8.03	$111,575.00

Again, for simplicity's sake, we will assume the same usability engineering plan with its associated cost as shown in Table 3.2. The project usability engineer estimated that for this project the usability engineering plan will produce a new site design with the *predicted benefits every month* summarized in Table 3.26.

Comparing these benefits and costs, the project usability engineer argued that the proposed usability engineering plan will pay for itself in the first 8 months after launch, as shown in Table 3.27, with a net benefit of $111,575.00 in the first year, and monthly benefits of $28,125.00 thereafter.

Note that an alternative net benefit calculation is used in this example. Instead of estimating benefits on an annual basis and then computing a net benefit for both the first year and for the expected application lifetime, in this case we predict benefits on a monthly basis and then divide the total cost of the usability engineering plan by the total predicted monthly benefits to determine a "payoff period," that is, the point at which the predicted benefits equal the cost. This is simply an alternative way to express the net benefit. After that point, net benefits are predicted to accrue on a monthly basis in the amount of the total predicted monthly benefit.

Below is a step-by-step description of how the project usability engineer arrived at the final results.

1. Start with the Usability Engineering Plan

Again, we follow the usability engineering plan laid out in Table 3.2.

2. Establish Analysis Parameters

In the case of an *e-commerce site*, the critical parameters are usually *volume of visitors* and *profit margin per purchase*. If the profit margin per online purchase is low, then a very large number of additional purchases would have to result from usability alone for the usability engineering costs to payoff. On the other hand, if the profit margin per online purchase is high, then only a small number of increased purchases must result from improved usability to pay for the usability engineering plan. Thus, these critical parameters will directly determine how much can profitably be invested in usability.

The analysis parameters for this sample project are summarized in Table 3.28. Again, some are the same or similar to those for the previous sample analyses, while some are different because of the use of different benefit categories (e.g., current average visitors per month).

Table 3.28 Analysis Parameters for an E-Commerce Web Site

Analysis Parameters	*Values*
Current average visitors per month:	125,000
Current buy-to-look ratio:	2%
Current rate of usage of the "Call Back" button:	2%
Profit margin per unit:	$10
Average length of servicing each use of "Call Back" button (3 minutes expressed as hours)	0.050000
User fully loaded hourly wage:	$25
Developer fully loaded hourly wage:	$175
Usability engineer fully loaded hourly wage:	$175
Manager fully loaded hourly wage:	$200
Customer support fully loaded hourly wage:	$50
Usability lab:	In place

3. *Calculate the Cost of Each Usability Engineering Lifecycle Task in the Usability Engineering Plan*

We are again simply using the usability engineering plan and associated costs given in Table 3.2 in this sample analysis.

The hourly rate for users used in this cost estimate is $25. Note that in this sample analysis, this is not based on a typical user's fully loaded hourly wage at their job, which it was in the case of a cost justification of traditional software development for internal users, or would be in the case of an analysis for intranet development for internal users. Instead it is based on the assumption that test users in the case of an e-commerce Web site will have to be recruited from the general public to participate in usability engineering tasks or techniques, and that they will be paid at a rate of $25 an hour for their time.

4. *Select Relevant Benefit Categories*

In this example, the project usability engineer decided to include the following benefits:

- Increased buy-to-look ratio
- Decreased abandoned shopping carts
- Decreased use of "Call Back" button

These benefit categories were selected because of their relevance to the audience for the analysis: the business sponsors of the site. There would undoubtedly be other very real potential benefits of the usability engineering plan in this case, but these were chosen for simplicity and to make a conservative prediction of benefits (as shown later). In particular, it should be noted that the benefit of decreased late design changes—included in the previous two examples—has been omitted in this example. This is simply because its benefit cannot easily be computed on a monthly basis, which all other benefits can in this case. If the net benefit calculations had been computed as a site lifetime benefit, as in the previous examples, rather than as a payoff period and first year benefit, then it could easily have been included, increasing the overall benefit. The usability engineer can point out this and other additional but omitted potential benefit categories to the relevant audience to argue that the real net benefit is actually likely to be even larger than the one presented, which is based on very conservative assumptions.

Comparing the new site design to the existing site design, the usability engineer anticipated that in the course of redesign, the usability engineering effort

would decrease abandoned shopping carts by insuring that the checkout process is clear, efficient, provides all the right information at the right time, and does not bother users with tedious entry of information they do not want or need to provide. He or she expected to improve the buy-to-look ratio by insuring that the right product information is contained on the site, and that navigation to find products is efficient and always successful. He or she also expected to decrease the use of the "Call Back" button by insuring that the information architecture matched users' expectations and by designing and validating a clear conceptual model, so that navigation of and interactions with the site are intuitively obvious. Accomplishing all these things depends on conducting the requirements analysis and testing activities in the proposed plan, as well as on applying general user interface design expertise.

5. Predict Benefits

Next the project usability engineer predicted the magnitude of each benefit that would be realized *if* the usability engineering plan (with its associated costs) is implemented. For example, he or she predicted how much *higher* the buy-to-look ratio would be on the site if it were re-engineered to be more usable than the existing site.

Benefit assumptions made in this analysis are given in Table 3.29. Benefit calculations based on the analysis parameters and these assumptions are given in Table 3.30.

The usability engineer based the benefit assumptions in this analysis on statistics available in the literature. In particular, he or she began with the often quoted average e-commerce Web site buy-to-look ratio of 2 to 3% (Sonderegger, 1998; Souza, 2000), then based the assumption that this ratio could be improved by a minimum of 2% (1% from improving the product search process, and 1% from improving the checkout process) through usability engineering tech-

Table 3.29 Benefit Assumptions for an E-Commerce Web Site[a]

Increased Buy-to-Look Ratio	*Decreased Abandoned Shopping Carts*	*Decreased Use of "Call Back" Button*
1% incr. in visitors who decide to buy (and checkout successfully)	1% incr. in visitors who have already decided to buy but who also now checkout successfully	1% decr. in visitors who use the "Call Back" button

[a] All percentages are relative to total monthly visitors.

Table 3.30 Benefit Calculations for an E-Commerce Web Site

Increased Buy-to-Look Ratio						
Current monthly visitors		Rate of increase in buyers		Profit margin per unit		Total
125,000	×	1%	×	$10	=	**$12,500.00**

Decreased Abandoned Shopping Carts						
Current monthly visitors		Rate of increase in buyers		Profit margin per unit		Total
125,000	×	1%	×	$10	=	**$12,500.00**

Decreased Usage of "all Back" button								
Current monthly visitors		Rate of decrease in use of "Call Back" button		# Hours saved per call eliminated		Customer support hourly rate		Total
125,000	×	1%	×	0.050000	×	$50	=	**$3,125.00**

niques, based on a variety of statistics available in the literature. For example, Souza (2001) suggests that it is typical for as many as 5% of online shoppers to fail to find the product and offer they are looking for, and cites one study in which 65% of shopping attempts at a set of prominent e-commerce sites ended in failure. Sonderegger (1998) suggests that sales underperform on e-commerce sites by as much as 50% or more because of poor site usability. GVU (1999) survey data suggests that almost 50% of Web site users cannot find the information they are looking for, and that over 80% of Web shoppers have left one site for another when they had dissatisfying experiences with site usability. Souza (2001) also noted that (at the time he interviewed for his report) companies spent between $100,000 and $1,000,000 to redesign specific sites, but few had any sense of which specific design changes might have paid off.

The project usability engineer based the assumption of reduced usage of the "Call Back" button by 1% on statistics suggesting that as many as 20% of e-commerce site users typically call in to get more information (Souza, 2001.)

Most of us have experienced all these problems—difficulty finding products, difficulty checking out, and need to use a "Call Back" button to complete transactions—and would have little argument with the idea that they are typical. Given all the dramatic statistics cited, the modest assumptions made in this analysis

seem very conservative indeed. And given the fact that companies typically spend a great deal of money on redesign with no process in place that can ensure improvements in usability, the notion of a highly structured and goal-oriented usability engineering process starts to make a lot of sense.

6. Compare Costs to Benefits

The net benefit calculations for this analysis were given in Table 3.27. In this example, it can be seen that the expected payoff period is 8 months, and that after that, benefits are predicted to continue to accrue at a rate of $28,125 per month. Because the new site is expected to have a lifetime much longer than 8 months, the project usability engineer expected the usability engineering plan to be approved based on this cost justification.

Given the very clear net benefit, the project usability engineer would be wise to stick with this fairly aggressive plan and submit it to project management for approval.

If the estimated payoff period had been long, or if there were no reasonable payoff period, then the team would be well advised to go back and rethink the plan, scaling back to shortcut techniques for certain tasks, and perhaps collapsing the design process from three to two or even one design level, to reduce the costs. However, even with very conservative benefit assumptions, a short payoff period is predicted. Benefits could very likely be even more robust than those predicted, further shortening the payoff period and increasing the ongoing accrual of benefits after the payoff period.

3.4.4 A Product Information Site

This example is based on a hypothetical scenario given in a Forrester report from June 2001 called "Get ROI from Design" (Souza, 2001). It involves an automobile manufacturing company that has put up a Web site that allows customers to get information about the features of the different models of cars they offer and options available on those cars. It allows users to configure a base model with options of their choice and get sticker price information. Users cannot purchase a car online through this Web site—it is meant to generate leads, and points users to dealerships and salespeople in their area.

Again, for simplicity's sake, we will assume the same usability engineering plan with its associated cost as is shown in Table 3.2. The project usability engineer estimated that for this project the usability engineering plan will produce a new site design with the *expected benefits every month* summarized in Table 3.31.

Table 3.31 Expected Monthly Benefits for a Product Information Web Site

Benefit Category	*Benefit Value* per Month
Increased lead generation	$37,500.00
Total monthly benefit	**$37,500.00**

Table 3.32 Net Benefit Calculations for a Product Information Web Site

Benefits/Month	*Total Cost*	*Payoff Period in Months (Cost ÷ Benefit)*	*Net Benefit (First Year)*
$37,500.00	$225,925.00	6.02	$224,075.00

Comparing these benefits and costs, the project usability engineer argued that the proposed usability engineering plan would pay for itself in the first 6 months after launch, as shown in Table 3.32, and that after that period benefits would continue to accrue at a rate of $37,500 per month. Because the new site is expected to have a lifetime of more than 6 months, the project usability engineer expected the usability engineering plan to be approved based on this cost justification.

Note that again, instead of estimating benefits on an annual basis and then computing a net benefit for both the first year and for the expected application lifetime, in this case we predict benefits on a monthly basis, and then divide the total cost of the usability engineering plan by the total predicted monthly benefits to determine a "payoff period," that is, the point at which the benefits accrue to equal the cost. After that point, net benefits are predicted to accrue on a monthly basis in the amount of the total monthly benefit each month.

1. Start with the Usability Engineering Plan

In this example, we again start with the same assumed plan as in all previous examples, presented in Table 3.2.

2. Establish Analysis Parameters

Analysis parameters for this example are presented in Table 3.33. Again, we are assuming there is an existing site with known traffic statistics, and that the project involves a redesign.

Table 3.33 Analysis Parameters for a Product Information Web Site

Analysis Parameters	*Values*
Current average visitors per month:	250,000
Current percent of visitors that result in a concrete sales lead:	1%
Current percent of leads generating a sale	10%
Profit margin per sale:	$300
User fully loaded hourly wage:	$25
Developer fully loaded hourly wage:	$175
Usability engineer fully loaded hourly wage:	$175
Manager fully loaded hourly wage:	$200
Customer support fully loaded hourly wage:	$50
Usability lab:	In place

3. Calculate the Cost of Each Usability Engineering Lifecycle Task in the Usability Engineering Plan

We will use the same cost calculations as in the previous examples, shown in Table 3.2.

4. Select Relevant Benefit Categories

Since this is a *product information site,* only certain benefit categories are of relevance to the business goals of this redesign project. The project usability engineer decides to include just one benefit category in the analysis: increased lead generation.

The project usability engineer selected this benefit category because he or she knows it will be of most relevance to the audience for the analysis: the business sponsors of the site. There may be other potential benefits of the usability engineering plan (such as decreased late design changes), but the usability engineer chose just this one for simplicity and to make a conservative prediction of benefits.

As compared to the existing site design, the usability engineer anticipated that in the course of redesign, the usability engineering effort would increase leads by ensuring that visitors can find basic information and successfully configure models with options. Achieving this will depend on conducting the requirements analysis and testing activities in the proposed plan, as well as on applying general user interface design expertise.

5. Predict Benefits

Next the project usability engineer predicted the magnitude of the benefit that would be realized *if* the usability engineering plan (with its associated costs) were implemented. In this case, he or she predicted how much *higher* the lead generation rate would be on the site if it were re-engineered for usability. Table 3.34 presents the benefit assumption made in the analysis, and Table 3.35 presents the benefit calculations, which incorporate both the benefit assumption and analysis parameters.

The project usability engineer makes the very conservative assumption that a better interface will result in 1% more leads each month. Traffic statistics from the current site show that considerably more than 1% of configuration attempts are currently abandoned, representing lost opportunities for leads. To support the benefits estimate, the usability engineer points out existing design flaws that might account for abandoned configuration attempts and that could be rectified in the redesign process.

6. Compare Costs to Benefits

Next the usability engineer compared benefits and costs to determine the payoff period. This was shown in Table 3.32.

Again, the project usability engineer's initial usability engineering plan appears to be well justified. It was an aggressive plan, in that it included all Lifecycle tasks and used the more reliable and thorough techniques for each task.

Table 3.34 Benefit Assumptions for a Product Information Web Site

Increased Lead Generation
1% incr. in visitors who generate a lead

Table 3.35 Benefit Calculations for a Product Information Web Site

Increased Lead Generation								
Current Monthly Visitors		Increase in Visitors Who Generate a Lead		Percent of Leads that Generate a Sale		Profit Margin/ Sale		Total
250,000	×	1%	×	10%	×	$300	=	**$37,500.00**

In addition, the benefit assumptions were conservative. Given the short estimated payoff period, he or she would be wise to stick with this aggressive plan and submit it to project management for approval and funding.

3.5 SUMMARY

The particularly critical value of usability to the ROI of Web sites and Web-enabled applications is illustrated by the following case study.

A contract development team was building a Web site for a client organization. The Web site was to include up-to-date drug information and was intended to be used by physicians as a substitute for the standard desk references they currently use to look up drug information such as side effects, interactions, appropriate uses, and data from clinical trials. The business model for the site was an advertising model. Physicians were expected to visit the site regularly because it offered more current and more easily found information than the published desk references (such as The Physician's Desk Reference, or PDR). The marketing plan was to have pharmaceutical companies buy advertising for their drug products on the site because the visitors to the site (physicians) represented their target market. Regular and increasing traffic from repeat visitors, and new visitors joining based on word-of-mouth amongst physicians, would drive up the value of advertising, generating a profit—and ROI—for the client.

The development team generated a prototype design that the client would use to pursue venture capital to support the full-blown development and initial launch and maintenance of the site. The client paid for this prototype development.

Once the prototype was ready, a usability engineer was brought in to design and conduct a usability test. Eight physicians were paid to fly to the development center and participate in usability testing. Several basic search tasks were designed for the physicians to perform. They were pointed to the prototype's homepage, and then left on their own to try to successfully find the drug information that was requested in the first task.

Within 45 seconds of starting their first search task, *seven out of the eight physicians gave up,* and announced, unsolicited, that the site was unusable and that if it were a real site, they would abandon it at that point and never return.

Clearly, if the site had launched as it was designed prior to this test, not only would an optimal ROI not have been realized, but in fact, the site would have failed altogether and a complete loss of the clients' investment would have resulted. If seven eighths of all visitors never returned, the Web site would not

have generated enough traffic to have motivated companies to buy advertising. The entire investment would have been lost.

Instead, the test users were asked to continue with the entire test protocol, and the data generated revealed insights into the problem that was a show stopper on the first task, as well as other problems uncovered in other test tasks. The site was redesigned to eliminate the identified problems. Clearly, the usability test, which had an associated cost, was worth the investment in this case.

This true anecdote illustrates something that distinguishes Web sites from commercial software products. In a commercial software product, the buyers discover the usability problems only after they have paid for the product. Often they cannot return it once they have opened the shrink-wrapped package and installed the product. Even if it has a money-back guarantee, returning it takes some effort and they are not likely to do so, especially if there are not many alternative products on the market with noticeably greater usability.

On a Web site, on the other hand, it costs the visitor nothing to make an initial visit. On a Web site based on an advertising model, such as the one just described, the site sponsor makes nothing at all unless there is sufficient *ongoing traffic* to attract and keep advertisers. On an e-commerce site, the sponsor makes nothing at all unless the visitors actually find and successfully *purchase products*, and unless usage of the Web channel increases sales or reduces the costs of other sales channels.

A Web site is not a product and the user does not have to buy it to use it. The Web site is just a channel, like a TV show, magazine, or catalog, and if users do not find and repeatedly and successfully use the channel, the investor gets no ROI for having developed the channel. Thus, usability can make or break the ROI for a Web site even more so than for traditional software products.

It is also true that success competing in the marketplace is even more dependent on relative usability on Web sites than is the case with traditional software products or sales channels. Someone wishing to buy a book may be inclined to buy from a particular brick-and-mortar bookstore that is easy to get to, even if it is not the best bookstore around. On the other hand, if customers cannot easily find the desired book through, for example, the barnesandnoble.com Web site, they need not even get out of their chairs to shop at a competitor's site instead—for example, amazon.com. It is not enough to simply have a Web site that supports direct sales; your site must be more usable than the competition's site (as well as have equivalent or superior content), or business will be lost based on the relative usability of the selling channel alone. For example, 60% of a sample of consumers shopping for travel online stated that if they cannot find what they are looking for quickly and easily on one travel site, they will simply leave and try a competitor's site (Harteveldt, 2000).

In addition, if you are a catalog order company such as L.L. Bean or Land's End, and your product is good but your Web site is bad, customers will not use your Web site and will resort to traditional sales channels (fax, phone) instead. This will result in a poor ROI for the Web site that was intended to be justified by relatively low operating costs compared with more traditional catalog sales methods.

Site usability is the equivalent of good—or *great*—customer service. Consider a company's current level of investment in traditional customer service channels. They need to make an equivalent investment in the usability of a site meant to replace or augment traditional channels of customer service. Site usability, like good traditional customer service, ensures that customers can find what they want to buy—an obvious prerequisite to sales. Just as salespeople in stores help you find the book you want, a good user interface on a bookseller's Web site must make browsing and searching easy and successful. Usability, like good customer service, also reduces errors in business transactions and the corresponding costs to fix those errors. For example, just as a good catalog offers accurate pictures of clothing and size charts to minimize the costs of processing returns, a clothing Web site must also ensure that customers don't order clothing in the wrong size or color to minimize returns. This might entail providing multiple views of the garment and sizing charts. In addition, usability, like good customer service, motivates customers to choose to use a Web site over traditional methods of doing business, ensuring an ROI in the Web site itself. Usability helps ensure that customers will return repeatedly to a Web site, just as good salespeople help ensure that customers will return to brick-and-mortar stores—again contributing to ROI. And usability done right in initial development is always cheaper than fixing usability problems identified after launch—or going out of business.

By contrast, lack of usability on an e-commerce site is the equivalent of poor customer service. Imagine having the following telephone conversation with a human customer service representative (CSR):

CSR:	Hello. Thank you for calling XYZ Shopping; how can I help you?
Shopper:	Hello, I would like to place an order.
CSR:	Do you know the name and extension number of the order taker?
Shopper:	Of course not! I just want to order something! Can't you take my order?
CSR:	No, you need to know who to ask for.

Much later, after finally finding out how to reach the order taker, then being put on hold repeatedly, and finally completing giving the required order informa-

tion, imagine that this customer has the following conversation with customer service:

Shopper: Oh, I just realized I need to make a change—can you do that?

CSR: If you don't know the name and extension of the order changer, you will just have to wait until your order arrives, then send it back and reorder . . .

Would *you* order by phone with this company again after such an experience? Unfortunately, this hypothetical interaction is analogous to many online shopping experiences. E-commerce Web sites too often make it very difficult to figure out how to place an order and make changes in midstream, and there are often long periods of being "on hold," waiting for graphics-heavy pages to download.

Thus, there are real risks associated with *not* investing in Web usability. When customers are unsatisfied with the quality of customer service on Web sites, customer loyalty erodes. Even companies with well-established brand loyalty may find this loyalty beginning to decline with the launch of an unusable Web site meant to replace traditional customer service channels. Customer dissatisfaction may result from an unacceptable learning curve to accomplish desired goals (as in the example of the physician's drug reference Web site cited earlier), from unacceptable task times necessary to accomplish desired goals (e.g., too many clicks or download times that are too long), or from an unacceptable rate of errors and confusion during task completion (e.g., abandoned shopping carts). Potential sales may be lost because customers can't find what they want to buy or get the information they need to make buy decisions. Customers may make errors in business transactions that cost time and money to rectify, and create customer dissatisfaction. For example, a vendor recently agreed to pay return shipping costs because the Web site misled one of the authors (Mayhew) into ordering an item twice. The second author (Tremaine) was charged California sales tax on a product shipped to New Jersey because the "out of state" button was placed in a nonobvious place on the screen. Both authors will think twice about using these Web sites again because of the time lost in repairing the needless errors. Customers may remain loyal but refuse to use a Web site and return to traditional methods of doing business, reducing the ROI on the Web site investment. Customers may also defect to competitor companies whose Web sites are more usable. Costly rework of a site to fix problems discovered after initial launch will also eat into the ROI.

The four cost-benefit analysis examples offered in this chapter are based on simple subsets of all actual costs and potential benefits and very simple and basic assumptions regarding the value of money over time. More complex and sophis-

ticated analyses can be calculated (see Karat, Chapter 4). However, usually a simple and straightforward analysis of the type offered in the examples above is sufficient for the purpose of winning funding for usability engineering investments in general, or planning appropriate usability engineering programs for specific development projects.

The sample cost-justification analyses offered here suggest that it is usually fairly easy to justify a significant investment of time and money in usability engineering during the development of software applications. The framework and examples presented in this chapter and elsewhere in this volume should help you demonstrate that this is the case for your software development project, Web-based or otherwise.

(Portions of this chapter are excerpted or adapted from Mayhew and Tremaine [1994], Mayhew [1999], and Mayhew and Bias [2003]. Used with permission.)

REFERENCES

Bias, R. G., and Mayhew, D. J. (1994). *Cost Justifying Usability.* Chestnut Hill, MA: Academic Press.

Bias, R. G., Mayhew, D. J., and Upmanyu, D. (2003). Cost Justification. In J. Jacko, and A. Sears (Eds.), *The Handbook of Human-Computer Interaction.* NJ: Lawrence Erlbaum Associates.

Dray, S., and Mrazek, D. (1996). A Day in the Life of a Family: An International Ethnographic Study. In D. R. Wixon, and J. Ramey (Eds.), *Field Methods Casebook for Software Design.* New York City: John Wiley & Sons, Inc.

GVU. (1999). www.gvu.gatech.edu/user_surveys. Atlanta, GA: Georgia Institute of Technology.

Harteveldt, H. H. (2000). Travel Data Overview. Cambridge, MA: Forrester Research.

Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. (1992). *Object-Oriented Software Engineering.* Harlow, England: Addison-Wesley.

Karat, C. M. (1989). Iterative Usability Testing of a Security Application. *Proceedings of the Human Factors Society 33rd Annual Meeting* (pp. 273–277). HFES.

Landauer, T. K. (1996). *The trouble with computers: Usefulness, usability, and productivity.* Cambridge: MIT Press.

Mantei, M. M., and Teorey, T. T. J. (1988). Cost/benefit for incorporating human factors in the software lifecycle. *ACM Communications,* 31 (4), 428–439.

Mayhew, D. J. (1999). *The Usability Engineering Lifecycle.* San Francisco: Morgan Kaufmann Publishers.

Mayhew, D. J., and Tremaine, M. M. (1994). A Basic Framework for Cost-Justifying Usability Engineering. In R. G. Bias, and D. J. Mayhew (Eds.), *Cost justifying Usability.* Boston: Academic Press.

Mayhew, D. J., and Bias, R. G. (2003). Cost-Justifying Web Usability. In J. Ratner (Ed.), *Human Factors and Web Development* (2nd ed.). NJ: Lawrence Erlbaum Associates.

Nielsen, J. (1993). *Usability Engineering.* Boston: Academic Press.

Perkins, R. (2002). Remote Usability Evaluations Using the Internet: Real Time vs. Instrumented and Automated Methods. *Proceedings of the 1st European UPA Conference* (pp. 79–86), September.

Snyder, C. (2003). *Paper Prototyping.* San Francisco: Morgan Kaufmann Publishers.

Sonderegger, P. (1998). The Age of Net Pragmatism. Cambridge: Forrester Research.

Sonderegger, P. (2000). Scenario Design. Cambridge: Forrester Research.

Souza, R. K. (2000). The Best of Retail Site Design. Cambridge: Forrester Research.

Souza, R. K. (2001). Get ROI From Design. Cambridge: Forrester Research.

Vaananen-Vainio-Mattila, K., and Ruuska, S. (2000). Designing mobile phones and communicators for consumers' needs at Nokia. In E. Bergman (Ed.), *Information Appliances and Beyond.* San Francisco: Morgan Kaufmann Publishers.

Whiteside, J., Bennett, J., and Holtzblatt, K. (1988). Usability Engineering: Our Experience and Evolution. In M. Helander (Ed.), *Handbook of Human-Computer Interaction.* Amsterdam: North-Holland.

Wixon, D., and Wilson, C. (1997). The Usability Engineering Framework for Product Design and Evaluation. In M. Helander, T. K. Landauer, and P. Prabhu (Ed.), *Handbook of Human-Computer Interaction* (2nd ed.). Amsterdam: North-Holland.