eful to the Na-
king of our ap-
asis on HCI in
Mei Xu skillful-

iance the rich-
pert advice re-
ftware for the
haer generous-
or Chapter 13.
nd Cascading
these matters.
Web accessibili-
tudy of a Web
d "bugs." Deb-
ridge, Jim Fu-
Bret Kroll, Lai
varez, Worawit
n Starkey con-

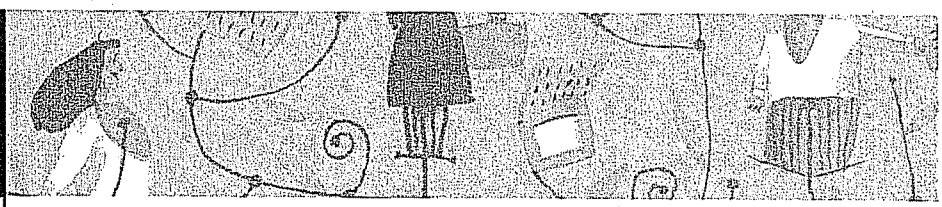
hitful and con-
n, Karl Flores,
llen, John Tap-

MaryJo David-
n Reiser at the
, Erika Rogers
niversity; Mary
seph's College.
rmer, Andrew
Connie Hey-
filley, Barbara
, Douglas R.
y P. Williams,

our editor at

D. McCracken
New York

salee J. Wolfe
Chicago



1 Human–Computer Interaction: An Overview

1.1 Introduction

Have you ever been annoyed by how hard it is to find something on the Web? Have you ever wondered why you had to give out your email address just to get into a site? Perhaps you have filled out an order form only to be sent to a page that says, "Sorry, you made a mistake – fill it out again." Have you waited on hold for hours to talk to someone in tech support because the instructions were lousy?

It doesn't have to be this way. Websites can be convenient and pleasant to experience when designers and developers take time to think about the users.

Goals of this Chapter

In this chapter, you will learn

- the benefits of making a website more usable
- the history and goals of human–computer interaction, the discipline that provides the tools for enhancing Web usability
- The methodology of user-centered development, which is essential for developing usable websites.

1.2 Benefits of Usable Websites

Since the 1993 introduction of NCSA Mosaic, the first point-and-click Web browser [Andreessen 1994], Internet use has exploded. However, in a period of less than 10 years, the novelty of using the Web has worn off. At first, just being innovative was enough for success; now, however, sites must meet user expectations in order to survive. Think about your experiences: if you visit a site and find it frustrating, what are the chances that you will go there again? On the other hand, websites that do meet user expectations enjoy many advantages. Four of the most important advantages are gaining a competitive edge, reducing development and maintenance costs, improving productivity, and lowering support costs [Donahue 1999].

1.2.1 Gaining a Competitive Edge

It's not the number of hits a site receives that matters; it's what people do once they get there that counts. Did they actually purchase something? Did they fill out the registration form? Essential to the success of a website is its *conversion rate*, which is the percentage of visitors who take action on the site. Taking action can mean

"buy something," if it's an online shopping site; it can mean "register to receive information," if it's a travel information site. For a personal website, it might mean using the "sign my guest book" feature. The average conversion rate for shopping sites is somewhere between 3 and 5 percent.

Suppose your company spends $2,500 on advertising that generates 5,000 visits to its shopping site. This sounds like a pretty good deal on the surface – your site is receiving a lot of hits. Suppose, though, that the conversion rate is only 2 percent. With a little bit of calculation, we can find the number of purchases:

$$2\% \text{ of } 5{,}000 \text{ visits} = 100 \text{ purchases.}$$

Divide 100 purchases into $2500 and each purchase costs $25 in advertising! But if the conversion rate is just a little higher, say 4 percent, then in the same 5,000 visits your site will attract 200 purchases, and the cost per purchase drops to $12.50. The higher the conversion rate, the better the sales and the greater the profit margins [Gurley 2000]. IBM experienced this phenomenon when they rolled out their redesigned "Shop IBM" website. In the first month, hits went up 120 percent, but sales increased 400 percent [Battey 1999].

What drives conversion rates? The top factor for high conversion rates is ease of use. Usable websites consistently have the highest conversion rates. Studies have found that, if customers have an enjoyable experience, they are likely to spend more time on a site, make purchases, and return to the site for further shopping. If they have to waste time searching for an item or figuring out how to buy it, they quickly become frustrated and leave. All other things being equal, the site that offers the better user experience will win the market place. This holds true even when the more usable site charges a slightly higher price [Rhodes 2001].

1.2.2 Reduced Development and Maintenance Costs

Learning about the needs of real users before creating a website results in lower development costs by saving you from implementing features that people don't want. It can also save you from costly corrections that can crop up after site rollout. According to one study [Donahue 1999], over half of all software life-cycle costs occur during the maintenance phase. Most maintenance costs arise from "unmet or unforeseen" user requirements.

1.2.3 Improved Productivity

For people using a shopping site, improved productivity means being able to purchase items quickly. For a company's intranet, it can mean that employees are able to complete their work more efficiently. At Bay Networks, a subsidiary of Nortel, company officials estimated that their improved intranet will allow each member of the sales staff to save a minimum of two minutes per day when searching for documents. Two minutes may seem like a small amount, but over a year's time, this translated into a savings of over 10 million dollars [Fabris, 1999].

1.2.4 Lower Support Costs

When a website is understandable, users don't need to call customer support. This can add up to significant—even huge—savings, given that some experts estimate that the cost of a single service call ranges between $12 and $250 [Donahue 1999].

In 1999, Wal-Mart decided to shut down their shopping site for three weeks rather than leave it online until they could roll out the new redesign. The fact that they decided that it

would be more cost effective to shut down the site, thus closing off any possibility of a sale, indicates that customer confusion was severely affecting the company's support systems [Weiss 2000].

1.3 What is HCI?

What makes a website usable? How is it possible to create a site that meets expectations? If your company decides that its current site needs improvement, how do you go about achieving it? How do you know whether the changes actually constitute an improvement? The answers to these questions are found in a discipline closely linked to computer science, called human–computer interaction—HCI, for short.

To understand HCI, it's helpful to know a little bit about the story of how it came to be. Its beginnings occurred slightly after the introduction of the computer. When electronic computers first appeared in the 1950s and 60s, they were extremely expensive. In 1966, a major midwestern university bought a computer called a Control Data CDC 6600. This particular computer had a 1-megahertz clock and less than 1 megabyte of memory. Its capacities were certainly modest when compared with today's computers, which typically have 2-gigahertz clocks and 512 megabytes of memory. Even so, the university spent $3,000,000 for this machine. Moreover, although $3,000,000 today is still a respectable sum, in 1966 it was worth a lot more. At the time, the average price of a house in the U.S. was $13,000, and the median salary was $7,436 [US 1968]. Compared to the cost of the computer, salaries were cheap. At the time, it made good economic sense to train experts to accommodate the computer. Little thought was given to the idea of making life easier for people using the programs.

Things began to change in the late 1970s and early 80s. Computers became smaller and cheaper. In 1981, IBM Corporation introduced the first IBM Personal Computer, targeted for home and small-business use. Because these machines cost less money, more people were able to purchase them. A new phenomenon occurred: nonexperts began using computers. For this new and rapidly growing group of people, computers were similar to cars and telephones: they were just tools to assist them in their work and lives. These people already had skills and knowledge in other areas, such as business and medicine. For them, knowledge of a computer's internal workings was not interesting, and they viewed learning this information as a waste of time.

Computer and software manufacturers noticed this trend and started considering the benefits of creating products that were "user-friendly." "If our product is easy to use," they reasoned, "then people will buy it." However, they didn't know any effective ways to discover what made a product user-friendly or how to design a product that *was* friendly. What was clear was that most programmers did not have good insights into these issues.

At the same time, researchers began studying these problems, drawing on previous work in human factors and ergonomics. The study of human factors had its origins in World War II efforts to design equipment that facilitates optimal human performance even under the most difficult circumstances, such as a combat situation. The field called *ergonomics* studies similar problems, but usually in a workplace setting. In 1982, the Association for Computing Machinery (ACM) approved the naming of a Special Interest Group on Computer–Human Interaction (SIGCHI), whose goals include promoting the use of human factors in the human–computer interaction process [Borman 1996].

One of this SIG's activities was to develop a definition for human–computer interaction. Here it is [ACM 1992]:

Human–computer interaction is a discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.

An interactive computing system might be a single PC equipped with a monitor, mouse, and keyboard that's capable of displaying Web pages; it might be an embedded device such as certain components of today's cars; it might be a WAP (Wireless Access Protocol)-enabled device, such as a cell phone; or it might be software that allows hundreds of people from around the world to collaborate on a single project. The major phenomena surrounding interactive computing systems include accommodations for physical limitations and considerations of the environment where the system is located. A system in a noisy hotel lobby will have requirements different from these of a system located in the quiet and privacy of a user's home.

On any development team, the role of an HCI practitioner is to serve as an advocate for the users of the system. An HCI practitioner listens to users and communicates their needs to the developers responsible for implementation. Some have likened the role to that of a family therapist, attempting to provide clear avenues of communication among users and developers [Kreitzberg 2000].

1.4 Goals of HCI

What does HCI try to accomplish? HCI strives to make people's experience with computers more productive, more time-efficient, and more pleasant. Eason gives a concise answer when he states the following:

The goals of HCI are to develop or improve the

- safety
- utility
- effectiveness
- efficiency
- usability
- appeal

of systems that include computers [Eason 1988].

Safety can mean "safety of users," "safety of data," or both. Safety of users is a primary consideration in systems such as air-traffic control and the monitoring equipment in a hospital's intensive care unit. Safety of data includes protection of files from unintentional or malicious tampering and issues such as privacy and security for websites.

Utility has to do with the services that the system provides. *Effectiveness* concerns a user's ability to accomplish a desired goal or to carry out work. Utility and effectiveness are distinct aspects of an interactive system. A website might provide all the information, instruction, and server-side support required to complete a purchase, which are all examples of utility. However, if users can't figure out how to find the items they want to buy, then the site lacks effectiveness.

Efficiency is a measure of how quickly users can accomplish goals or finish their work when using the system. *Usability* includes ease of learning and ease of use, and *appeal* describes how well users like the system, including such considerations as first impressions and long-term satisfaction.

The priority among these six aspects will vary, depending on the type of system or website you are creating. As mentioned in the previous paragraph, safety would be an overriding concern for air-traffic control systems. Another example would be monitoring software for nuclear power plants. Can you think of interactive systems where safety is less of an issue?

1.5 User-Centered Development Methodology

The user-centered development methodology is essential for developing successful user interfaces. A *user interface* comprises "those aspects of the system that a user comes in contact with" [Moran 1981]. The methodology outlined here is useful for creating any sort of interface, from spreadsheet programs to video games to websites. This development methodology differs from the more traditional software engineering methodologies in three key areas:

1. User-centered development is user centric, not data centric. It involves users in the process as much as possible with the goal of creating an interface that meets user expectations. This may include such activities as observing users while they work, inviting users to participate on the design team, and asking users to try out the product and following up on their feedback.
2. User-centered development is highly interdisciplinary and draws on knowledge from a multitude of areas, including art, psychology, technical writing, and computer science, among others. Figure 1–1 demonstrates the variety of disciplines that contribute to HCI.
3. The methodology is highly iterative and involves as much testing and revision as possible. In this book, you will see techniques to test and "debug" an interface before it is implemented, thus avoiding costly revisions once the interface has been coded.

The following is an overview of the user-centered development methodology. The initial stages involve the gathering of information; the later stages involve the designing, building, and testing of a prototype of the interface.

1.5.1 Needs Analysis

Needs analysis summarizes the nature and purpose of the interactive system you plan to develop. It describes the type of system—is it a website, a video game, or a spreadsheet? It mentions the people the system will serve and the benefits it will provide. This is very brief, typically no more than a couple of sentences that explain why it is a good idea. Here is an example:

> The Woods Bay website will promote tourism for Woods Bay Cottages in Bamfield, British Columbia. It will describe the accommodations, surroundings, fishing, snorkeling, hiking, and other local attractions for potential visitors. In addition, it will provide information about traveling to Bamfield, current rates, and booking a cabin.

Here is another:

> The redesigned website for the Plains Art Council in Burkmere, South Dakota will foster greater participation in the arts in the local community by providing a comprehensive

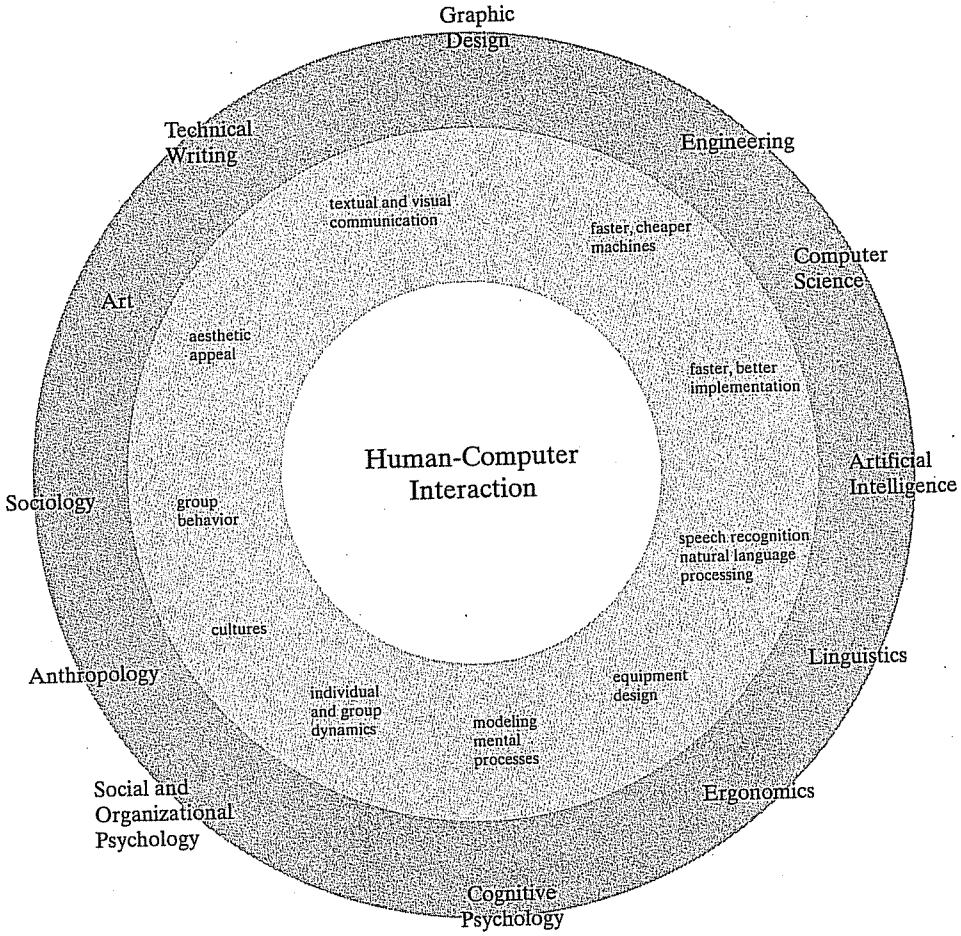


Figure 1–1 Disciplines that contribute to HCI.

listing of available art programs and services and by creating a virtual gallery or display space for local artists.

In the first example, the interactive system is a website and the audience is people who are planning a vacation to British Columbia. The site will tell them about the town and what to do there and will promote the hotel accommodations. In the second, who is the audience?

1.5.2 User and Task Analysis

User analysis characterizes the people who will use your site. This includes some general considerations, such as age, education level, and experience with computers. You will also look into users' experience and expectations with sites similar to yours. *Task analysis* looks at the type of work users will do at your website. It examines user *goals*, which are what users want to accomplish when visiting a site like yours. It also looks at *tasks*, or the activities they carry out to achieve their goals. If your site is an entertainment site, these "tasks" could actually be part of some sort of leisure activity.

Too many products fail because the development team didn't take the time to find out who their users are or what they want to do. A thorough user and task analysis will put you ahead of the game when it comes time to create your site. Chapter 3 will discuss techniques for conducting user and task analyses.

1.5.3 Functional Analysis

In a *functional analysis*, you identify the functionality—the computer services that users will need to carry out their tasks. For instance, for a travel site, tasks could include finding all of the flights to a particular location during a specified time range, ordered by price. To help users complete this task, your site will need a search function and a sorting capability. For a site selling music CDs, one of the tasks will be "Buy a CD." In this case, you will need secure on-line transaction functionality. Usually, there is a close correspondence between functions and tasks.

In this stage, you also decide what aspects of the tasks should be automated and which should be completed by humans. For example, on the Woods Bay Cottages site, the developers did not create an automated reservation system for the cottages. Instead, the site lists a toll-free phone number. In contrast, many major hotel chains allow you to book rooms directly on their sites. Access to computing resources and availability of skilled developers will influence decisions about what to automate.

1.5.4 Requirements Analysis

A *requirements analysis* describes the formal specifications required to implement any system, including websites. Depending on the application, the formal specification can include data dictionaries, entity–relationship diagrams, and object-oriented modeling. These are the same types of specifications that you would develop in a systems-analysis or software-engineering course. This particular step is well covered in other branches of the computing discipline, so this text will not cover it.

1.5.5 Setting Usability Specifications

You will need to answer the question, "How good is your website?" Setting usability specifications will help you determine this. Usability specifications include *performance measures*, such as "number of tasks completed" and "number of errors," which are directly observable user behaviors. Usability specifications also include *preference measures*, which give insights into a user's opinion about your site. Examples of preference measures include "first impression" and "overall satisfaction." Chapter 3 will discuss usability specifications and how to set them.

1.5.6 Design

In this step, you decide on the organization and appearance of your website. When designing, you identify the content for your site and organize it according to your users' expectations. In Chapter 4, you will learn techniques for organizing content, which is one of the most critical elements for success of a website. It's during the design step that you also decide on your site's look and feel. Design also includes the layout of individual pages and how to use visual organization techniques to create clarity and consistency between pages. Chapter 5 gives some straightforward, practical advice about how to lay out a Web page even if you have never had an art class. During the design phase, you will also decide how to set up the navigation, which is the topic of Chapter 6.

It's only at this stage that you begin sketching page layouts, because you now know who your users are and what they want to do. The natural temptation is to jump the gun and begin designing screens at an earlier stage, but, without knowing your users and their tasks, the results will not be satisfactory.

1.5.7 Prototyping

The word *prototype* comes from the Greek word *proto*, meaning first, and the word *type*. Thus, a prototype is an original model or a pattern. During prototyping, you create the model from which the website will (later) be implemented. You could decide to prototype the entire site, in what is called *global prototyping*, or you could prototype only selected parts of the site, using what is called *local prototyping*.

Prototypes can also be classified as *evolutionary* or *throw-away*. If the prototype becomes part of the final project, it's evolutionary. Throw-away prototyping is exactly what the term implies: the prototype serves only as a pattern for implementation, and you actually throw away the prototype once the site is complete. An additional classification of prototypes is *high fidelity* or *low fidelity*. High-fidelity prototypes closely resemble the final product in appearance and functionality. In contrast, a low-fidelity prototype would never be mistaken for a final product. Figure 1–2 shows an example of a low-fidelity and a high-fidelity prototype of a Web page.

There is a wide range of techniques and tools available for prototyping websites. Chapter 7 will discuss these in detail.

1.5.8 Evaluation

In the evaluation step, you test your prototype. In a very real sense, this is similar to testing a program. You need to know where the problems are in your prototype. There are two types of evaluation, *user-based evaluation* and *expert-based evaluation*. In a user-based evaluation,

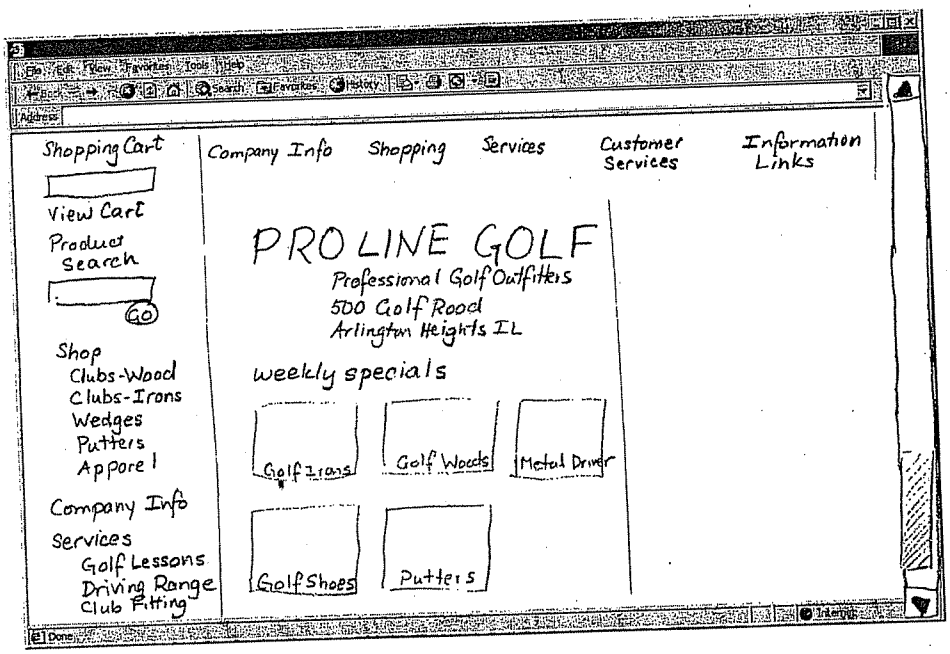


Figure 1–2 Low- vs. high-fidelity prototypes. Courtesy of Kirsten Pielstrom.

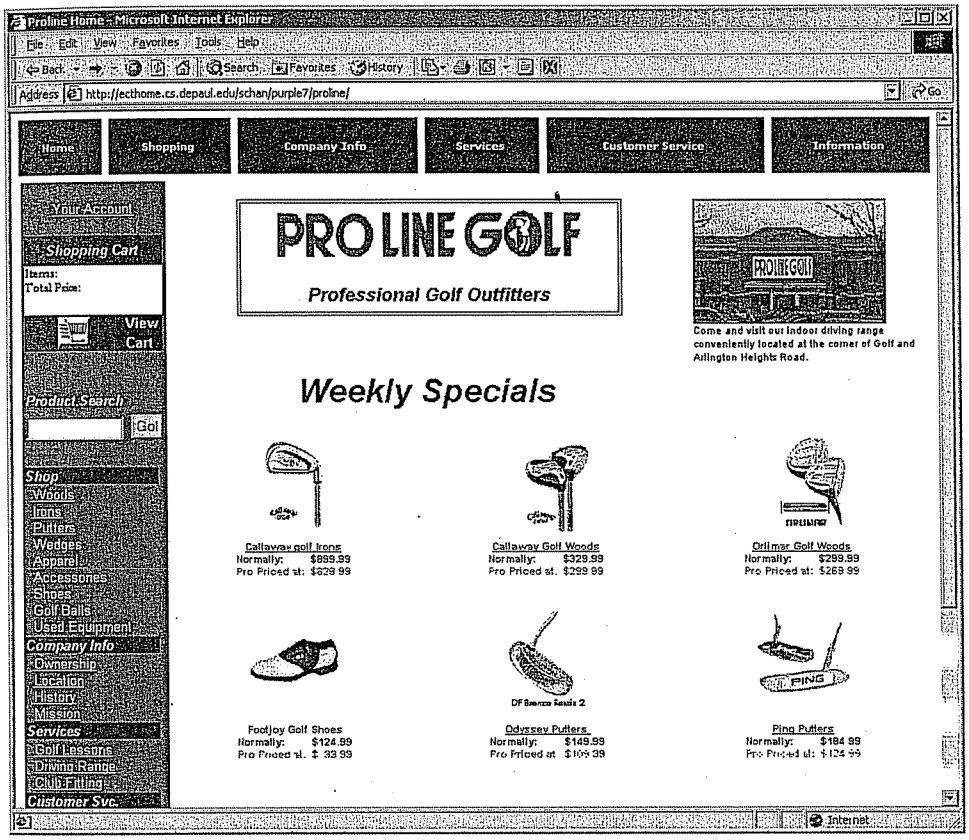


Figure 1–2 *(Continued)*

you ask actual users to perform representative tasks with your prototype. As a user carries out each task, an observer takes notes on where problems occurred. In an expert-based evaluation, a group of usability experts critique the prototype. Both approaches have their benefits, but this text will emphasize user-based testing. You will learn how to conduct user-based testing in Chapter 8. Given the results of your evaluation step, you will redesign sections of your prototype and test again.

1.6 Characteristics of User-Centered Development

User-centered development is highly iterative. There is a cycle of repetition in the design, prototype and evaluation steps of the process, as you can see in Figure 1–3. Ideally, you continue the design–prototype–evaluate cycle until you have met all of your usability specifications. The process is similar to programming, where you develop and debug programs via a cycle of design–code–test. You know that the program is working correctly when the actual outputs match the predicted outputs. Similarly, you know that your prototype is satisfactory when you meet the usability specifications.

While this book will be looking at how HCI can be used to create more effective Web sites, the principles and methodology mentioned here can be applied to any interactive system.