13

Usability Assessment

Overview

We have all experienced difficulty using everyday things. Video recorders, mobile phones, and microwave ovens are technological miracles, when you think about it, but they aim to help us perform relatively simple tasks. So, why aren't they easier to use? Door handles, can openers, pushchairs, and food packaging may seem relatively basic in terms of their technology, but these mundane products confuse, perplex, and frustrate us when they appear impossible to use. Donald Norman's book, *The Design of Everyday Things* [1], provides many examples of usability failures and explains the challenges of and possible routes towards user-centered design.

Since the first computers were built, much of the mystique surrounding the software systems that run on them stems from the fact that software is rarely intuitive. Many people work for companies that have implemented difficult-to-use applications. Anyone observing the people at supermarket checkouts, hotel reception desks, or the next desk at his or her own office will have seen and heard frustration expressed at the system being used. If a company implements an unusable system, its employees will probably get on with their jobs, but they are likely to be less efficient and comfortable than should be the case. But these are captive users. Put an unusable site onto the Web, and your users will not stick around very long. Your users have the freedom to abandon your site and visit your competitors'. It may have been possible to implement difficult-to-use systems in the past, but it is no longer.

One of the key selling points for Web sites is their ease of use, and Table 13.1 presents the risks addressed by usability assessment.

A *usability fault* is a potential problem in the appearance or organization of a system that makes it less easy for users to use. Operationally, a

Table 13.1
Risks Addressed by Usability Assessment

ID	Risk	Test Objective	Technique
U1	Use of bleeding-edge technologies is excessive or inappropriate.	Verify that sophisticated plug-ins, page objects, or design techniques are not used where they add no value or where simpler methods would work just as well.	Collaborative usability inspection
U2	Site privacy statement is missing or inaccurate.	Verify that privacy guidelines are observed and that an accurate privacy statement exists on the site.	Collaborative usability inspection
U3	Screen appearance is adversely affected by changes to resolution and window resizing.	Verify that Web pages are readable on monochrome devices, low-resolution monitors, and in small windows.	Collaborative usability inspection
U4	Site uses frames in an inappropriate way.	Verify that frames are not used (if proscribed by requirements) or that frames do not adversely affect usability.	Collaborative usability inspection
U5	Text layout and formatting problems make the page unpleasant, confusing, or too long.	Verify that text layout is logical, consistent, and understandable, not fragmented or compiled into overly large pages.	Collaborative usability inspection
U6	Graphics problems make the page unpleasant, confusing, clichéd, or pretentious.	Verify that graphics are relevant, sympathetic to the appearance of the site, correctly sized, appropriate to anticipated users, and in keeping with the purpose of the site.	Collaborative usability inspection
U7	On-line user documentation and help is of poor quality.	Verify that on-line documentation is accurate, relevant, concise, and useful.	Collaborative usability inspection
U8	Users find the system time-consuming, error-prone, and inconsistent.	Demonstrate that selected tasks can be performed efficiently, accurately, and reliably.	Collaborative usability inspection

Table 13.1 (continued)

ID	Risk	Test Objective	Technique
U9	Site navigation is difficult to understand, illogical, or disorganized and obstructs the user.	Demonstrate that the navigation through pages is intuitive, consistent, and makes the user feel in control.	Usability test
U10	Site lacks a personalization facility (or the facility does not work).	Demonstrate that users may personalize the site to work in their preferred manner without being intrusive and without capturing personal data unnecessarily.	Usability test
U11	Site or pages are not accessible to people with disabilities.	Verify that accessibility conventions are adhered to.	Web accessibility checking
U12	Incorrect or poorly written text on pages confuses or misleads users.	Verify that all text, messages, and help screens, as well as the information provided, are accurate, concise, helpful, and understandable.	Previously covered by content checking (Chapter 9)
U13	The HTML page or objects embedded in the page are so large that the page is too slow to download.	Verify that objects are small enough to download within an acceptable time.	Previously covered by object load and timing (Chapter 10)
U14	The system cannot meet response time requirements while processing the design loads.	Demonstrate that the entire technical architecture meets load and response-time requirements.	Previously covered by performance and stress testing (Chapter 12)

usability fault is any clear or evident violation of an established usability principle or guideline; or it is any aspect of a system that is likely to lead to confusion, error, delay, or failure to complete some task on the part of the user. Usability faults have two dimensions:

1. The location and identity of a fault;
2. The problem and rationale for identifying it.

Applying the user-interface design principles set forth in Constantine and Lockwood's book, *Software for Use* [2], some examples of Web usability faults are summarized in Table 13.2.

Usability and usability assessment have never been as important as they are now. Typical Web site users have low boredom, frustration, inconvenience, and insecurity thresholds. Home-based users may never have been trained in the use of a computer, let alone browsers and your Web applications. Regardless of the user's level of sophistication, if your Web application doesn't allow enquiries to be made and orders to be placed easily, quickly, and reliably, the site will fail. If the user cannot easily understand from your Web site how to proceed, the user will leave your site and go elsewhere.

The first two methods of usability assessment discussed here, collaborative usability inspection and usability testing, are derived from the work of Larry Constantine and Lucy Lockwood in their book *Software for Use* [2]. We are grateful to the authors and publishers, Addison-Wesley, for allowing us to adapt their material for this chapter. For a more thorough discussion of the method and related references, you should refer to this work. Their Web site at http://www.foruse.com has numerous other resources on usability.

In his presentation to the User Interface '99 Conference [3], Larry Constantine identifies six methods for improving Web usability categorized as either expensive in the long term or most effective (Table 13.3).

Of the three methods deemed most effective, usability inspections stand out as the easiest and cheapest to implement, and early inspection is more efficient that later testing. This is consistent with all studies of inspection versus dynamic testing methods. Usability testing has its place, and we promote that as a method for addressing the specific areas of usefulness, ease of navigation, and personalization. Usage-centered design is, of course, an

Table 13.2
Some Examples of Usability Faults and Their Classifications

Location: Identity	**Rationale: Problem**
Personalization page: there is no cancel button	Tolerance principle: forced selection if dialogue opened mistakenly
Corporate home page: navigation links are unclear	Simplicity principle: user has to click on links to see where they lead
All pages: the hundreds of links on home page appear on all pages; required button is not easily found.	Visibility principle: the continue button is hidden among hundreds of other links

Table 13.3
Comparison of Methods for Improving Web Usability

Method	Expensive in the Long Term	Most Effective
Rapid iterative refinement: build something, try it, build it again	Yes	
Style guides: standard style and good guidelines		
Usability testing: actual trials, real users, no demos	Yes	Yes
Usability design metrics: hard numbers from designs and sketches		
Usability inspections: taking an informed, critical look		Yes
Usage-centered design: building in usability from the start		Yes

Source: [3]. Reproduced with permission.

approach to the overall design, implementation, and assessment of user interfaces, and its principles should be promoted in all development projects. It is not specifically a test activity.

Web accessibility checking, if done manually, might be deemed to be part of usability inspection as it could be performed at the same time as other checking. Accessibility guidelines, however, have been prepared, and it is now possible to perform the checking of a Web page against these guidelines using automated tools. Because it can be automated, Web accessibility testing has been described as a distinct test type in this chapter.

Other Methods

Beta testing is a commonly used method for gaining feedback from interested potential customers or friendly existing customers. Although it appears to be an inexpensive technique for the system supplier to employ, the feedback obtained is highly colored by the fact that usually only a self-selecting minority of beta testers respond. Even if all your beta testers give feedback, you have no control over their personal agendas or preferences. It is common for beta testers to blame product usability problems on themselves and figure out alternative ways of doing things. Usability problems may never come to light if this happens. Beta testing has the benefit of being cheap, but it can also be ineffective or misleading.

There are a number of other recognized techniques for usability assessment. Cognitive walkthroughs [4] are a detailed review of a sequence of steps that an interface requires a user to perform in order to accomplish some task. The reviewers check each step for potential usability problems. Usually, the main focus of the cognitive walkthrough is limited to understanding how easy a system is to learn. Questionnaire-based methods, such as SUMI [5] and WAMMI [6], collect subjective evaluation data and analyze it to product statistical scores on key attributes of the system under evaluation.

Collaborative Usability Inspection

Collaborative usability inspection is a systematic examination of a finished product, design, or prototype from the point of view of its usability by intended end users. The process is a team effort that includes developers, end users, application or domain experts, and usability specialists working in collaboration. A major benefit of the method is its own usability: It is easy to learn. Experienced inspection teams have been known to detect 100 usability faults per hour. The collaborative nature of these inspections means that users and developers understand the relationship between user interactions and design constraints and decisions. These inspections can be performed at any stage of development from the assessment of prototypes to the finished Web site, but, of course, the cost of fault correction increases the later the inspections take place. Collaborative usability inspection draws from other inspection techniques, most notably heuristic evaluation, a technique devised by Jakob Nielsen and succinctly described on his Web site at http://www.useit.com [7]. This same Web site also has a wealth of Web-related usability articles, papers, and guidance.

Heuristics on Which to Base Inspections

Inspections are guided by a set of rules or heuristics for good user-interface design. These are used as a framework for identifying and categorizing usability faults. Constantine and Lockwood [2] propose a set of five usability rules and six generic user interface design principles reproduced in Tables 13.4 and 13.5.

Nielsen promotes a (popular) set of 10 Web design heuristics [8] summarized in Table 13.6. His book, *Designing Web Usability* [9], is packed with usability design guidelines from which you can extract your own preferred rules if you care to.

Table 13.4
Five Usability Rules

Rule	Description
Access	The system should be usable, without help or instruction, by a user who has knowledge and experience in the application domain, but no prior experience with the system.
Efficacy	The system should not interfere with or impede efficient use by a skilled user who has substantial experience with the system.
Progression	The system should facilitate continuous advancement in knowledge, skill, and facility and accommodate progressive change in usage as the user gains experience with the system.
Support	The system should support the real work that users are trying to accomplish by making it easier, simpler, faster, or more fun or by making new things possible.
Context	The system should be suited to the real conditions and actual environment of the operational context within which it will be deployed and used.

Source: [2]. Reproduced with permission.

Table 13.5
Six Usability Design Principles

Principle	Description
Structure	Organize the user interface purposefully in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things, and making similar things resemble one another.
Simplicity	Make simple, common tasks simple to do, communicating clearly and simply in the user's own language and providing good shortcuts that are meaningfully related to longer procedures.
Visibility	Keep all needed options and materials for a given task visible without distracting the user with extraneous or redundant information.
Feedback	Keep users informed of actions or interpretations, changes of state or conditions, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
Tolerance	Be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonably.
Reuse	Reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

Source: [2]. Reproduced with permission.

Table 13.6
Usability Heuristics

1.	Visibility of system status
2.	Match between system and the real world
3.	User control and freedom
4.	Consistency and standards
5.	Error prevention
6.	Recognition rather than recall
7.	Flexibility and efficiency of use
8.	Aesthetic and minimalist design
9.	Help for users to recognize, diagnose, and recover from errors
10.	Help and documentation

After: [8].

Other rules can be used, and several collections of checklists are available. See [10–12] for substantial checklists of usability design issues that could be employed, but probably need to be summarized to be useful as inspection rules. One of the benefits of using heuristics is that the rules can be distributed between the inspection participants to spread the workload a little—it is difficult for anyone to keep an eye on 10 heuristics at once.

Vincent Flanders and Michael Willis's book, *Web Pages That Suck* [13], takes a different approach to learning good design—studying bad design. Their book is a good source of practical recommendations that could also be used as a checklist.

Approach

Inspection is a pure fault-finding activity and its objectives are to detect usability faults and user-interface inconsistencies. The inspection team should adopt the tester mentality and act like impatient, intolerant, and difficult users. Inspection is intensive work and requires the team to be focused, persistent, and nitpicking. Somewhat like proofreading a book, the inspection demands that both the superficial appearance and underlying structure of the system be examined. Having users on the team makes it easier for the team to become more user-focused.

Because the developers are involved in the inspection it can be a little uncomfortable for them to see their work pulled to pieces by others. The traditional view of testing applies: Finding faults is good. Faults found can be fixed and the system improved. Celebrating faults and mocking the developers are not helpful. Faults should be discussed in technical terms with a view to getting the developers to accept them and understand their impact so they can formulate a better design. Where the system exhibits highly usable or well-designed features, these should be noted to ensure the developers do not change them when the faults are corrected.

When faults are reported, the developers should not dispute the faults, criticize the users, or get defensive, which might suppress the team's willingness to report faults. When a reviewer gets stuck, the developers should not suggest alternative methods of completing the desired task. This would hide the fact that the reviewers are lacking some information, and that is a fault. It is not helpful to discuss or argue the pros and cons of a particular fault: The team should focus on finding and logging faults as quickly as possible. Some faults may eventually be deemed so trivial that they are not fixed, but this decision to act and change the system is made by the stakeholders.

Inspection Roles

The members of the inspection team fulfill distinct roles. Teams can be as small as 3 or 4 people or as large as 20, but 6 to 12 is the optimum range of team size. The common roles allocated to inspection team members are summarized in Table 13.7.

Preparation for the Inspection

The lead reviewer briefs the other reviewers and organizes the room, equipment, servers, network for the inspection. This might include obtaining large monitors or projection screens so all reviewers can see the system in use and compiling paper copies of all Web pages under test or having a printer nearby to capture hardcopy versions of problem pages. The reviewers should not prepare before the inspection meeting so they can approach the inspection with a fresh outlook. With the help of the users and developers, the lead reviewer should assemble a collection of scenarios or test cases (see an example below) to be exercised in the inspection. It is important to set aside enough time to conduct the inspections, but it is also important not to let them drag on for too long. Systems that cannot be inspected in less than 3 or 4 hours need to be inspected over multiple sessions, each lasting a half day or less. The inspection will be performed in two phases.

Table 13.7
Usability Inspection Roles

Role	Description
Lead reviewer	Convenes and leads the inspection. Manages the team; ensures the process is followed; keeps the team focused. Contributes to the inspection, but does not dominate.
Inspection recorder	Maintains the records of faults found and their classification. Assesses and records the severity of the faults. Lead reviewer or continuity reviewer may fulfill this role in small teams.
Continuity reviewer	Optional role. Focuses on gaps in continuity or inconsistencies. In some projects, this role may be delegated to the entire team. In other projects, a separate continuity inspection may be justified.
Developers	Act as dedicated reviewers. Forbidden to argue, challenge, defend, design, or promise anything to other reviewers. The lead reviewer enforces this discipline.
Users and domain experts	Genuine end users to act as reviewers. If not users, then involve domain experts who are asked to think and act like users. User-reported problems should be taken most seriously, and users should be encouraged to raise as many faults as possible.
Usability specialists	Act as consultants to facilitate and explore the faults raised by the reviewers. Do not act as arbiters of fault severity.

Interactive Inspection Phase

In the first phase of inspection, the system is used to perform selected tasks. These tasks are defined prior to the meeting, but not scripted like a test script. Rather, they define a series of scenarios, such as the following:

> Register as a new user. Search for a book entitled *Web Page Testing* and buy it online using your credit card (number 9999 9999 9999 9999). Read the confirmation e-mail sent by the Web site and, using the e-mail instructions, obtain the status of your order.

Larry Constantine recommends, "The Lead Reviewer or someone else should drive the system in response to user statements about what they would do next. This is also the only practical way to deal with multiple users who have divergent approaches … you should recommend against having more than two users in an inspection—too chaotic" (personal communication).

Comments from users or any of the other reviewers should be noted if a fault is detected. All of the reviewers should be looking for problems. The lead reviewer should prompt the team to note any observations in the following instances:

- When a new page appears (and note any immediately observable problems);
- At the end of each step or stage in the scenario;
- At the end of the scenario to summarize the views of the team.

Of course, the reviewers may make observations at any time. The lead reviewer facilitates the generation of comments and controls the flow of the inspection to allow the recorder to keep up with the logging. Observations on usable and effective features are also logged to ensure the developers do not change things that work well.

Static Inspection Phase

In the second phase, the inspection team visits as many of the pages in as many contexts as possible and examines each page in finer detail. It is probably impossible to visit all pages via the various navigation paths through the system, so the lead reviewer needs to manage the scope of this activity to ensure as many pages are inspected as possible in the time available. At this point, the reviewers should examine the details of all images and other embedded objects, the graphical detail on the page, form fields, buttons, drop-down lists, radio buttons, and check boxes to make sure there is no conflict with the usability rules.

Follow-Up

At the end of the inspection, the inspection team should review the fault list that has been logged by the inspection recorder. The faults should be evaluated by the inspection team and project manager (for a small project) or may be submitted to a project acceptance team on larger projects. The decision to be made is, does this fault make the system unacceptable?

The cost of correction (and the impact of change) is an important consideration and where the potential impact is less severe, these faults might not be fixed at all. This is ultimately a decision for the stakeholders—some faults may be deemed less severe and deferred to later releases. Faults need to be allocated a severity in line with their impact in production, then corrected by the developers.

Other Inspections

Two variations exist on the standard inspection method. Consistency inspections are required where there are particular concerns over the consistency of various pages that users access in their interactions and also within the content of individual pages. Selected groups of features are inspected for their consistency, and typical groupings you might use could include the following:

- Menu bars and buttons;
- Dialog boxes;
- Data fields and text labels;
- Error messages.

The important issue with consistency inspections is to be able to compare the contents of each group. Pinning hardcopy screenshots of Web pages or dialog boxes to the wall and looking at them is the simplest method. The inspection is focused on inconsistencies, not other usability issues, because broadening the scope makes these inspections less efficient.

Usability Testing

Usability testing aims to detect usability faults in a system by using the system to perform selected tasks. There are two broad ways of performing usability testing: in a usability laboratory or in the field. We will only consider field usability testing here. For a broader description of both methods, see [2].

Usability testing of a finished system is effective for finding certain faults, but is no substitute for good usability design. Experience shows that usability tests uncover fewer faults than inspections, but the faults that are found are often more subtle or unusual in nature, and they can be more serious. Usability testing requires that a working system be available, so these tests tend to be possible only late in the project. If there isn't time to fix these faults, they may be left in the released product. Overall, usability tests are most effective when used to investigate specific areas of an application or to explore particular issues.

Testing in the Field

Testing in the field requires that the system be used in the workplace. Users are invited to use the system to do real work; they may be asked to perform selected tasks and, perhaps, to repeat those tasks (with variations) many times. At its most formal, field testing simulates testing in a laboratory with video and sound equipment, mixers, and monitors being used by the trained observers. At its least formal, the observer watches the user operate the system and takes notes. It is possible to ask the users to keep their own notes for later analysis, but this distracts the user from the task and may influence the results of the test. The benefit of field use is the greater realism and possibly the comfort factor for the users: They are at their normal place of work. Where users are acting like home-based personal users, you could let them work at home, but to observe them at home might be regarded as too intrusive. An alternative would be to get volunteer users to come to your site and to put them in as comfortable and relaxed an environment as possible to conduct the test. However formally the test is arranged, observers should take care not to influence or disturb the user testers. The way in which instructions are given or the body language displayed by observers during the test itself may have a detrimental effect.

Test Process

Usability tests in the field are usually organized, observed, and analyzed by the usability testers. The volunteer users (or subjects) perform the tests—they are asked to perform selected tasks using the system under test. The system under test need not necessarily be a finished product, but it must incorporate all the features required to perform the specified tasks. It could be an alpha or beta version, but should be stable enough not to crash during these tests.

A key discipline of such tests is that the testers should observe the tests, but not interact with the subjects in a way that unduly influences their behavior. In particular, the testers should not comment on the users actions, answer questions, or suggest alternative ways of completing the task. To guarantee this objectivity, the testers might set up a camcorder to record the subjects' activities and display the live pictures in another room. The testers can take notes based on the actions they witness and study the recordings for later analysis.

User-subjects can be encouraged to talk out loud while performing the tests. In this way, the thoughts that go through each user's mind are captured live, as the action happens. This provides a more complete reflection of the user's experience and thought processes while using the system under test. It is easy to imagine what is happening in the following scenerio:

> I've added the printer cable to my shopping basket, but the system tells me to register as a user before I proceed to the checkout. Register user…nope, can't find that button. Ahhh, perhaps the New Customers link is the one to try. Yes it is. Why didn't they say that before?

This narrative makes it obvious what the user is having difficulty with. If the user didn't record these comments, the few seconds lost finding the right command might never have registered as a problem in his or her mind. Even if the user had recorded the fact that he or she had difficulty, it might not occur to the developers reading the test log what the problem was. The developers would see what they wanted to see, of course, because they designed and built the screens in the first place. This approach is not infallible, however, because the subjects, knowing they are being recorded, may self-censor their comments or become self-conscious. They might suppress certain comments because they don't want to look slow or incompetent or to be seen as unsystematic or lacking in confidence. Some people might simply find it difficult to work on the system and comment on their own actions at the same time. The users may need some coaching and encouragement to speak their minds.

An alternative approach that has some merit is testing done in pairs. This requires one person to use the keyboard and mouse with another talking through the task and reading the documentation. Both users observe the system's behavior and comment on or discuss what they see on screen. Although this might be a more natural way to operate the system and comfortable way to extract comments, it does of course require two people rather than one, so the pace of testing might be reduced as a consequence.

One final alternative is to record the expressions of users while they perform the tasks. Although a user might not say anything, a perplexed expression might indicate that something is wrong. When the subject sees the replay of their face, they can usually recall what gave them difficulty. Arranging a mirror attached to the PC monitor could bring the users expression onto the same field of view as the screen, so the two can be seen and filmed together. Otherwise, a second video recorder would be required. The user would not be asked to provide a running commentary, so this usage would be closer to normal usage. After the session is over, the testers use the facial expressions as a trigger for questions like, You looked confused. What was going through your mind just then?

After the test, the users should be debriefed. This is mainly to obtain their impressions of the system, feedback on what worked well and not so well, and suggestions for improvement. When taken in the round, these observations may reveal trends that greatly assist with the diagnosis of problems and possible solutions.

Among other things, Web usability testing aims to expose difficulties that a user experiences when engaged in the following activities:

- Performing particular business transactions using the Web site (e.g., searching for and buying a book, locating and reserving a flight to an unusual destination);
- Navigating a Web site (probably to find information or access a particular resource);
- Personalizing a site to achieve the faster, more efficient browsing experience the personalization is meant to deliver.

Test cases or scenarios should be selected to cover any particular tasks of concern.

Web Accessibility Checking

It is possible to conduct manual inspections of a system to detect departures from user-interface standards or accessibility guidelines. In the context of the Web, accessibility refers to the ease with which anyone can make use of the Web, regardless of his or her technology, location, or disability. The most important accessibility guidelines for Web-based systems are defined as part of the Web Accessibility Initiative (WAI), whose stated mission is "… to lead the Web to its full potential including promoting a high degree of usability for people with disabilities." Their most important guidelines are the "Web Content Accessibility Guidelines" [14]. This document defines a comprehensive collection of guidelines to make your Web pages more accessible to users with disabilities, although the vast majority of guidelines would make a site more accessible (or usable) to all users. Consequently, these guidelines provide a ready-made checklist of points to inspect Web pages against. Guidance for evaluating Web sites for accessibility can be found in [15], which includes a step-by-step process for accessibility evaluation. A major benefit of these guidelines is that they can be used as objective requirements. Failure to adhere to these guidelines can be raised as incidents.

There are several assistive technologies now available to help users with disabilities use the Web. These include the following:

- Text-to-speech (TTS) browsers that use synthesized speech to read text on Web sites to a user;
- Text-only browsers that render Web sites in a text-only format;
- Voice-enabled browsers that navigate Web sites using speech commands;
- Specialized keyboards and mice;
- Voice-recognition software;
- Refreshable braille devices that transfer text onto a special device;
- Screen magnification or enhancement software.

Standard HTML has features that specifically support the use of these technologies and, as a consequence, feature strongly in the accessibility guidelines. One simple example is that images must have some associated or alternate text, should the user turn them off or be blind. This alternate text is used by voice-enabled browsers to provide the user with an audible narrative describing what is visible on the Web page.

These accessibility recommendations will become increasingly important as it is likely that some or all may become mandatory in some applications. Section 508 [16] is a U.S. federal mandate requiring that information technology be made accessible to people with disabilities. In 1998, the Workforce Investment Act established Section 508 as an amendment to the Rehabilitation Act of 1973. It requires that electronic and information technology developed, procured, maintained, or used by the federal government be accessible to people with disabilities.

Much of Section 508 compliance concerns making Web sites, intranets, and Web-enabled applications more accessible. This affects government agencies as well as private-sector institutions. Government agencies must make their Web sites and intranets accessible according to the standards and can only create or purchase Web-enabled applications that are compliant with 508 standards. The law impacts corporations doing business with the government because those corporations need to make their Web-enabled applications and any Web sites built for government clients accessible as well.

Fortunately, these guidelines can be verified using automated tools. The best known of these is produced by the CAST organization and is called

Bobby [17]. Bobby verifies Web pages against the WAI recommendations and reports anomalies. It can be used on-line or downloaded for use on your intranet. Automated verification of accessibility is possible to do at any point where a Web page is available to test, so it is probably a job that the developers perform.

Tools for Assessing Web Page Accessibility

Evaluating the usability of the human/computer interface is somewhat subjective, and the areas where tools can replace manual or visual assessment are limited. One particular area where tools do offer some support is in verifying accessibility. Because the majority of accessibility guidelines set out by the WAI can be related directly to the Web page's HTML, it is possible for automated tools to scan HTML and compare it with the accessibility rules. There are also some specific issues, such as color blindness, that can also be checked. Table 13.8 lists some of the proprietary and free tools that are available on the Web. For a more comprehensive listing, visit the W3C Web site at http://www.w3c.org.

Table 13.8
Tools for Usability Assessment

Proprietary Tools	**URL**
AccVerify	http://www.hisoftware.com/access/
InSight	http://www.ssbtechnologies.com
Colorfield Insight	http://www.colorfield.com
A User Friendliness Checklist	http://www.pantos.org/atw/35317.html
Lift	http://www.usablenet.com
Web Metrics Testbed	http://www.nist.gov/webmet
Vischeck	http://www.vischeck.com
Free Tools	**URL**
Tool to verify page on varying screen sizes	http://www.anybrowser.com
Bobby	http://www.cast.org/bobby/
WebSat, WebCat tools	http://www.nist.gov/webmet

References

[1] Norman, D. A., *The Design of Everyday Things*, New York: Doubleday, 1988.

[2] Constantine, L. L., and L. A. D. Lockwood, *Software for Use*, Boston, MA: Addison-Wesley, 1999.

[3] Constantine, L., "Web Usability Inspections," *Proc. User Interface '99 Conf.*, San Francisco, California, March 1999 (also available at http://www.foruse.com/).

[4] Wharton, C., et al., "The Cognitive Walkthrough Method: A Practitioner's Guide." In *Usability Inspection Methods,* pp.105–141, J. Nielsen and R. L. Mack (Eds.), New York: John Wiley & Sons, 1994.

[5] Human Factors Research Group, "Software Usability Measurement Inventory (SUMI)," University College Cork, Ireland, http://www.ucc.ie/hfrg/questionnaires/sumi/index.html, 2002.

[6] HFRG Ireland and Nomos Management AB Sweden, "WAMMI Web Usability Questionnaire," http://www.nomos.se/wammi/, 2002.

[7] Neilsen J., "How to Conduct an Heuristic Evaluation," http://www.useit.com, 2002.

[8] Neilsen, J., "Ten Usability Heuristics," http://www.useit.com/papers/heuristic/heuristic_list.html, 2002.

[9] Nielsen, J., *Designing Web Usability*, Indianapolis, IN: New Riders, 2000.

[10] Dustin, E., J. Rashka, and D. McDiarmid, *Quality Web Systems*, Boston, MA: Addison-Wesley, 2001.

[11] Nguyen, H. Q., *Testing Applications on the Web*, New York: John Wiley & Sons, 2001.

[12] Abeleto, Ltd., "Objective Evaluation of Likely Usability Hazards—Preliminaries for User Testing," http://www.abeleto.com, 2002.

[13] Flanders V., and M. Willis, *Web Pages That Suck*, San Francisco, CA: Sybex, 1998.

[14] World Wide Web Consortium (W3C), "Web Accessibility Initiative, Web Content Accessibility Guidelines 1.0," http://www.w3.org/TR/WCAG10/, 2002.

[15] World Wide Web Consortium (W3C), "Evaluating Web Sites for Accessibility," http://www.w3.org/WAI/eval/, 2002.

[16] Architectural and Transportation Barriers Compliance Board, "Electronic and Information Technology Accessibility Standards," http://www.access-board.gov/sec508/508standards.htm, 2002.

[17] CAST, "Bobby Web Page Accessibility Checker," http://www.cast.org/bobby/, 2002.