DOCTOR OF SCIENCES

Methodology for automating web usability and accessibility evaluation by guideline

Beirekdar, Abdo

Award date:
2004

Awarding institution:
University of Namur

Facultés Universitaires
Notre-Dame de la Paix

A Methodology for Automating Guideline Review of Web Sites

Abdo Beirekdar

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Sciences
(Computer Science Option)

- August 30th, 2004 -

Institut d'Informatique

NAMUR

Chapter 2

State of the Art of Automated Web Evaluation

2.1 Introduction

The World Wide Web has become the predominant means for communicating and presenting information on a broad scale. Unfortunately, Website usability and accessibility continue to be pressing problem [Ivory and al. 2003]. An estimated 90% of sites provide inadequate usability [Forrester Research 1999], and an estimated 66% of sites are inaccessible to users with disabilities [Jackson-Sanborn et al. 2002]. Although numerous assistive devices, such as screen readers and special keyboards, facilitate Website use, these devices may not improve a user's ability to find information, purchase products and complete other tasks on sites. For example, sites may not have links to help blind users skip over navigation bars, or sites may not enable users to increase the font size of text, so that they can read it.

A wide range of U&A evaluation techniques have been proposed and a subset of these techniques is currently in common use. Some techniques, such as formal testing, can only be applied after the interface design or prototype has been implemented. Others, such as heuristic evaluation, can be applied in the early stages of design. Each technique has its own requirements, and generally different techniques uncover different problems.

Usability and accessibility findings can vary widely when different evaluators study the same user interface, even if they use the same evaluation technique [Bailey et al. 1992; Desurvire 1994]. This result implies a lack of systematicity or predictability in the findings of UAE. Furthermore, UAE typically only covers a subset of the possible actions users might take. For these reasons, usability and accessibility experts often recommend using several different evaluation techniques [Dix et al. 1998; Nielsen 1993].

How can systematic way of producing results and fuller coverage in usability and accessibility assessment be achieved? One solution is to increase the number of teams evaluating the system, and to increase the number of study participants. An alternative is to automate some activities related to U&A evaluation.

This Automation would have several potential advantages over non-automated evaluation, such as:

- Increasing consistency of the errors uncovered. In some cases it is possible to develop models of task completion within an interface, and software tools can consistently detect deviations from these models. It is also possible to detect usage patterns that suggest possible errors.
- Increasing the coverage of evaluated features. Due to time, cost, and resource constraints, it is not always possible to assess every single aspect of an

interface. Software tools that generate plausible usage traces make it possible to evaluate aspects of interfaces that may not otherwise be assessed.

- Enabling comparisons between alternative designs. Because of time, cost, and resource constraints, non automated evaluation methods typically assess only one design or a small subset of features from multiple designs. Some automated analysis approaches, such as analytical modeling and simulation, enable designers to compare predicted performance for alternative designs.
- Predicting time and error costs across an entire design. As previously discussed, it is not always possible to assess every single aspect of an interface using non-automated evaluation. Software tools, such as analytical models, make it possible to widen the coverage of evaluated features.
- Reducing the need for evaluation expertise among individual evaluators. Automating some aspects of evaluation, such as the analysis or critique activities, could aid designers who do not have expertise in those aspects of evaluation.
- Reducing the cost of U&A evaluation. Automated methods can decrease the time spent on evaluation and consequently the cost. For example, software tools that automatically log events during usability testing eliminate the need for manual logging, which can typically take up a substantial portion of evaluation time.
- Incorporating evaluation within the design phase of UI development, as opposed to being applied after implementation. This is important because evaluation with most non-automated methods can typically be done only after the interface or prototype has been built and changes are more costly [Nielsen 1993]. Modeling and simulation tools make it possible to explore UI designs earlier.

It is important to note that automation is considered to be a useful complement and addition to standard evaluation techniques such as heuristic evaluation and testing -not a substitute. Different techniques uncover different kinds of problems, and subjective measures such as user satisfaction are unlikely to be predictable by automated methods.

Next, we will use the term usability evaluation (UE) instead of U&A evaluation because usability evaluation is more frequently sited in the Web evaluation literature, but same concepts apply to accessibility evaluation.

2.2 Usability evaluation: Basic Concepts

Below, after giving some general definitions, we focus on usability evaluation techniques.

2.2.1 Usability and Web Site Design Process

Creating a site that is of universal benefit to its users is a far more complex task than most new (and some experienced) web designers realize. Although it may be straightforward to code HTML pages, creating an effective site requires designers to follow practices, such as user-centered design or guideline application.

To gain insight about the process of creating sites, Newman and Landay [2000] conducted an ethnographic study of professional web site designers. They discovered that designers go through several design iterations with paper sketches and create different representations of a site —site maps depicting the site organization, storyboards depicting task completion steps, and schematics illustrating page layouts. Other literature sources (e.g., [De Troyer and Leune 1998; Ford and Boyarski 1997; Fuccella 1997; Sano 1996; Shneiderman 1997; van Duyne *et al.* 2002]) describe similar web site design and development processes. Figure 2.1 summarizes the phases that are often mentioned.

Site requirements are identified during the discovery phase; requirements need to address many issues, such as who the intended users are, what tasks they need to accomplish, what constraints need to be met, and so on. Several design alternatives may be explored during the design exploration phase. However, one design idea is chosen and improved upon during the design refinement phase. Site implementation typically occurs during the production phase; design templates or style guides may be developed initially to guide the implementation effort. The implemented site is tested and repaired, if necessary, during the quality assurance phase; afterwards, it is released for broad use. Once the site is released, it undergoes continuous maintenance (e.g., updating content). Iteration may occur during each of these phases, as depicted in Figure 1.1.

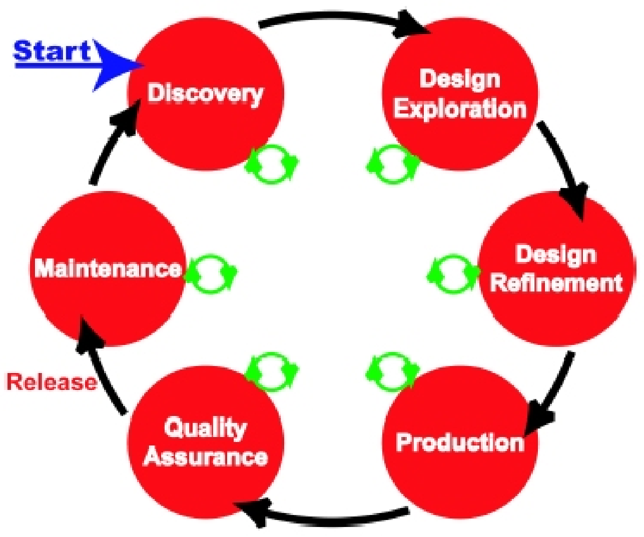


Figure 2.1: Web site design and development process.

To ensure that sites will be of universal benefit to their users, web designers may employ a number of practices during this design and development process. For instance, they may follow a usability engineering process (Figure 2.2) [Ivory 2003a].

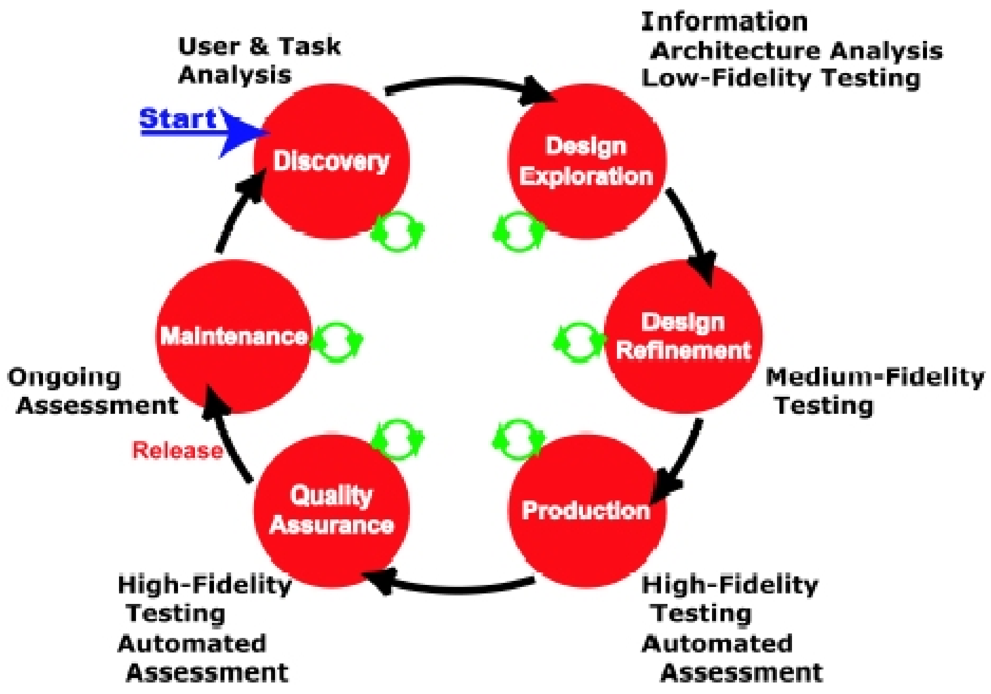


Figure 2.2: Usability engineering incorporated in the web site design and development process.

2.2.2 Usability Evaluation

Usability engineering [Nielsen 1993] concerns the development of systematic methods to support UE.

"Usability evaluation" can be defined as the act of measuring (or identifying potential issues affecting) usability attributes of a system or device with respect to particular context (users, performing particular tasks, in particular environments). The reason that context is part of the definition is that the values of usability attributes can vary depending on the background knowledge and experience of users, the tasks for which the system is used, and the environments in which it is used [Hilbert and Redmiles 2000].

UE consists of methodologies for measuring the usability aspects of a system's user interface (UI) and identifying specific problems.

"Usability data" is any information that is useful in measuring (or identifying potential issues affecting) the usability attributes of a system under evaluation [Hilbert and Redmiles 2000].

When evaluating a user interface, it is important to know what usability means for the current application. This is why an analysis of users and their needs is important: if we do not know what the user wants and needs, we cannot know which tasks s/he must be able to perform [Lecerof and Paternò 1998]. Evaluation can be performed at different times in the development process. During the early stages evaluations tend to be done to predict the usability of the product or to check the design team's understanding of the users' requirements. Later on in the design process the focus is more on identifying usability problems and improving the user interface.

2.2.3 Empirical and analytical approaches

Various types of approaches have been proposed for usability evaluation purpose. They can be substantially grouped in two approaches: empirical and analytical approaches.

In empirical testing the behavior of real users is considered. Thus, end users (extraneous to development) are involved in the evaluation process. It can be very expensive and it can have some limitations too. It requires long observations, which can take a lot of time to designers, and some relevant aspects can still be missed.

In analytical approaches, the evaluation is conducted by designers or expert evaluators. In these approaches, evaluators can use Model-based or inspection-based techniques.

Model-based approaches to usability evaluation use some models, usually task or user models, to support this evaluation. In inspection-based techniques for usability evaluation designers analyze a user interface or its description.

2.2.4 Evaluation phases

There are many techniques applied to evaluate user interfaces; some techniques can be applied during the development phase (e.g. heuristic analysis), while others have to be used only when the design or prototype is finished (e.g. formal analysis).

Evaluation involves several activities which depend on the method used. The activities more often used are:

- **Capture**: it consists of collecting usability data, such as task completion time, errors, guideline violations, and subjective ratings;
- **Analysis**: is the phase in which usability data are interpreted to identify usability problems in the interface;
- **Critique**: consists of suggesting solutions or improvements to mitigate the problems identified.

Many methods have been proposed because different techniques reveal different usability problems: usability evaluation typically only covers a subset of the possible actions users might take. Furthermore, different testers may detect different problems even if the same evaluation method is used. For these reasons, usability experts often recommend using several different evaluation techniques [Dix et al. 1998; Nielsen 1993]. Thus, further efforts (e.g. increasing the evaluator team) are required. One solution to this issue is trying to automate all, or part of the evaluation process activities (capture, analysis, critique).

2.2.5 Black-testing and White-testing

Referring to software testing in general, the classical distinction of test techniques is between *black-box* and *white-box* (pictorial terms derived from the world of integrated circuit testing). Test techniques are here classified according to whether the tests rely on information about how the software has been designed and coded (*white-box testing*), or instead only rely on the input/output behavior, without no assumption about what happens in between the “pins” (precisely, the entry/exit points) of the system *black-box*.

Therefore, the terms that software testers use to describe how they approach their testing are: black-box testing and white-box testing [Patton 2001].

In **black-box** testing, the tester only knows what the software is supposed to do (he can not look in the box to see how it operates): he types in a certain input, he gets a certain output. He does not know how or why it happens, just that it does.

In **white-box** testing, the software tester has access to the program’s code and can examine it for clues to help him with his testing (he can see inside the box). Based on what he sees, the tester may determine that certain cases are more or less likely to fail and can tailor his testing based on that information.

Since a Web application is just like any other software, these concepts can be applied to Web site testing.

The easiest starting point is treating the Web page or the entire Web site as a black-box. We do not know anything about how it works, we do not have a specification, we just have the Web site in front of us to test: There are all the basic elements-text, graphics, hyperlinks to other pages on the site, and hyperlinks to other pages play videos.

When planning the tests for a Web site, we take care to identify all the features of each page. For example, in order to satisfy usability properties, such as flexibility and efficient navigation, we have to check for text layout issues by resizing browser windows to be very small or very large (this will reveal design errors where the designer or the programmer assumed a fixed page width or height), or to check if the page has too many links or frames (contextual and efficient navigation properties).

Moreover, if we would like to consider accessibility problems, we can verify the presence of a link of textual version, the alternative texts for images and graphical links, etc.

So, we can say that black-box testing is achieved by user testing, in which users, or evaluators, use and observe directly the behavior of Web site, without considering the code of Web pages.

All examples that we have mentioned above are referred to static Web pages, but more and more the content of Web site are built dynamically. Thus, several features could be more effectively tested with a white-box approach. Of course, they could also be tested as a black-box, but the potential complexity is such that it is almost impossible to be sure to find the important design mistakes without some knowledge of the Web site’s system structure and implementation, such as the dynamic content, dynamically created Web pages, server performance and loading, etc.

In addition to HTML code (formed by tags only), a Web page can be created by using java, java script, cgi, perl, etc. For this reason, in order to better investigate usability and accessibility problems, it is required to examine the code of the Web site. To do that, we have to analyze this code through a white-box approach.

2.3 Usability evaluation taxonomies

There exist several taxonomies proposed to classify these methods and tools; Hom [1998] and Zhang [1999] provide a detailed discussion of inspection, inquiry, and testing methods (these terms are defined in Section 2.3.3). The most commonly used taxonomy is one that distinguishes between predictive (e.g. GOMS analysis and cognitive walkthrough) and experimental (e.g. usability testing) techniques [Coutaz 1995]. Another one is based on the presence or absence of a user and computer [Whitefield et al. 1991]. These two classification schemes, however, do not consider automation aspects.

2.3.1 Balbo [1995]

The first taxonomy which weighs up automation was proposed by Balbo. This classification distinguishes among four approaches to automation in dependence on which level the automation is applied. So, Balbo uses the following categories to classify 13 UE methods:

- **Non Automatic**: no level of automation supported (i.e., specialist evaluator performs method).
- **Automatic Capture**: software automatically captures interface usage (e.g., logging). Methods that use this automation rely on software facilities to record relevant information about the user and the system.
- **Automatic Analysis**: automatic identification of usability problems. These Methods are able to identify usability problems automatically.
- **Automatic Critique**: automatic analysis coupled with automated suggestions for improvements. Methods which not only point out difficulties but propose suggestions.

Balbo uses these categories to classify thirteen common and uncommon evaluation methods. However, most of the methods surveyed require extensive human effort, because they rely on formal usability testing or require extensive evaluator interaction. For example, Balbo classifies several techniques for processing log files as automatic analysis methods despite the fact that these approaches require formal testing or informal use to generate those log files.

What Balbo calls an automatic critic method may require the evaluator to create a complex interface model as input. Thus, this classification scheme is somewhat misleading, because it ignores the non-automated requirements of the evaluation methods.

2.3.2 Brajnik [2000]

Brajnik proposed a taxonomy that focuses on existing Web UE tools only. It classifies these tools according to:

- **Location**: web-based vs. off-line;
- **Type of service**:
    - *Failure identifiers*: they discover potential failures via simulation of user actions, like filling a form; sometimes they rank them according to severity.

  - *Fault analyzers*: they find failures and highlight their causes, i.e. faults; usually they systematically analyze the source code of the website; sometimes ranking the list of faults according to their severity.
  - *Analysis and repair tools*: they assist the developer also in fixing the faults.
- **Information source**: automatic usability analysis can be performed on the basis of the actual implementation of a website (sources), or on web server logs, or data acquired during user testing (user testing data); this paper deals only with tools analyzing website sources; and
- **Scope**: the set of attributes that are considered during the automatic analysis. A classification based on scope is:
  - *HTML validators and cleaners*: they assist in removing non standard usage of the language.
  - *HTML/graphic optimizers*: they improve downloading and rendering performance by recoding certain parts of HTML or graphic documents.
  - *Link checkers*: they probe all the links leaving a page to determine if their targets exist; and
  - *Usability tools*: they detect and sometimes help to fix usability faults.

2.3.3 Ivory and Hearst [2001]

Another more recent taxonomy is that proposed in [Ivory and Hearst 2001] and is continuously updated by Ivory. According to this taxonomy, UE methods are grouped along four dimensions: Method Class, Method Type, Automation Type and effort level.

- **Method Class**: the type of evaluation conducted at a high level. In this dimension UE methods can be classified into 5 method classes:
  - *Testing*: an evaluator observes users interacting with an interface (i.e., completing tasks) to determine usability problems.
  - *Inspection*: an evaluator uses a set of criteria or heuristics to identify potential usability problems in an interface.
  - *Inquiry*: users provide feedback on an interface via interviews, surveys, etc.
  - *Analytical Modeling*: an evaluator employs user and interface models to generate usability predictions.
  - *Simulation*: an evaluator employs user and interface models to mime a user interacting with an interface and report the results of this interaction (e.g. simulated activities, errors and other quantitative measures).

UE methods in the testing, inspection, and inquiry classes are appropriate for identifying specific usability problems, and for obtaining general assessments of usability. Analytical modeling and simulation are engineering approaches to UE that enable evaluators to predict usability with user and interface models.

Software engineering practices have had a major influence on the first three classes, while the latter two are quite similar to performance evaluation techniques used to analyze the performance of computer systems [Jain 1991].

- **Method Type**: How the evaluation is conducted within a method class. Method type indicates how the evaluation (capture and analysis) is conducted. For instance, if we use a user testing, we can record user actions by video types, or we can adopt remote evaluation by log files. Or, if we can proceed by inspection evaluation, we can use heuristics, or guidelines.
- **Automation Type**: the evaluation aspect that is automated. This dimension specifies which aspects of usability evaluation are automated, that is which phases are automated.
- **Effort level**: the type of effort required to execute the method. Even if we consider Balbo's automation taxonomy, a certain human effort is required to execute those non-automated phases of the method. In order to estimate needed effort, an attribute called effort level is added to each method. Thus, this attribute can assume the following values:
    - *Minimal Effort*: does not require interface usage or modeling.
    - *Model Development*: requires the evaluator to develop a UI model and/or a user model in order to employ the method.
    - *Informal Use*: requires completion of freely chosen tasks (i.e., unconstrained use by a user or evaluator).
    - *Formal Use*: requires completion of specially selected tasks (i.e., constrained use by a user or evaluator).

2.3.4 Methods for Web ergonomic evaluation

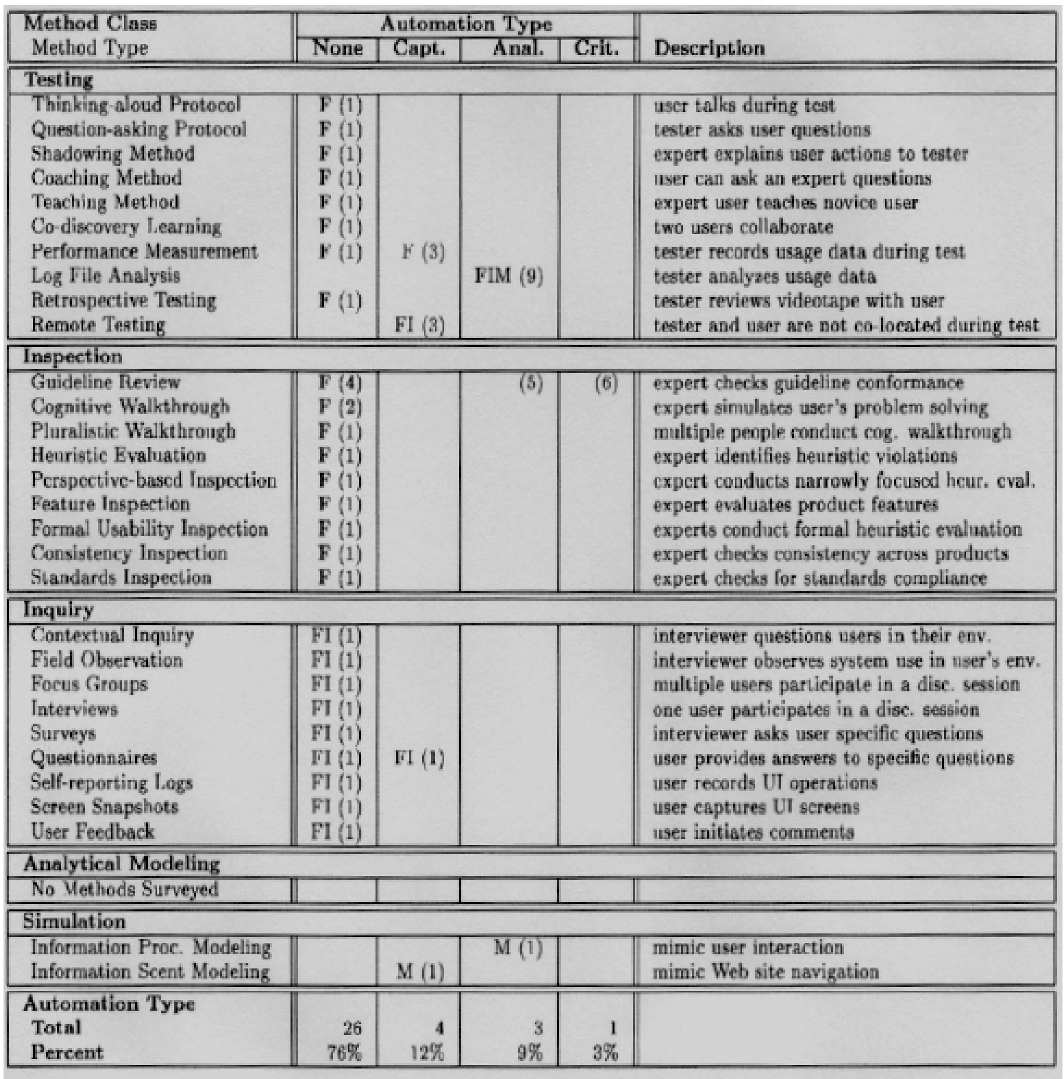
Ivory [Ivory and Hearst 2001] in her work surveyed 75 UE methods applied to WIMP interfaces, and 57 methods applied to Web UIs. Of these 132 methods, only 29 apply to both Web and WIMP UIs. Table 2.1 (taken from [Ivory and Hearst 2001]) shows that automation in general is greatly under-explored. Methods without automation support represent 67% of the methods surveyed, while methods with automation support collectively represent only 33%. Of this 33%, capture methods represent 13%, analysis methods represent 18% and critique methods represent 2%. All but two of the capture methods require some level of interface usage.

To provide the fullest automation support, software would have to critique interfaces without requiring formal or informal use. The survey revealed that this level of automation has been developed for only two method types: guideline review and cognitive walkthrough.

Of those methods that support the analysis evaluation table 2.1 shows that log file analysis, guideline review, and simulation methods represent the majority. Most of these methods do not require formal or informal interface use. However, they do require the site to be implemented.

2.3.5 Supported analyses

The existing tools support five general types of analyses: structural, HTML and CSS (cascading style sheet) coding, usability and accessibility, visual, and textual. We discuss these general approaches.



Method Class	Automation Type				
Method Type	None	Capt.	Anal.	Crit.	Description
Testing					
Thinking-aloud Protocol	F (1)				user talks during test
Question-asking Protocol	F (1)				tester asks user questions
Shadowing Method	F (1)				expert explains user actions to tester
Coaching Method	F (1)				user can ask an expert questions
Teaching Method	F (1)				expert user teaches novice user
Co-discovery Learning	F (1)				two users collaborate
Performance Measurement	F (1)	F (3)			tester records usage data during test
Log File Analysis			FIM (9)		tester analyzes usage data
Retrospective Testing	F (1)				tester reviews videotape with user
Remote Testing		FI (3)			tester and user are not co-located during test
Inspection					
Guideline Review	F (4)		(5)	(6)	expert checks guideline conformance
Cognitive Walkthrough	F (2)				expert simulates user's problem solving
Pluralistic Walkthrough	F (1)				multiple people conduct cog. walkthrough
Heuristic Evaluation	F (1)				expert identifies heuristic violations
Perspective-based Inspection	F (1)				expert conducts narrowly focused heur. eval.
Feature Inspection	F (1)				expert evaluates product features
Formal Usability Inspection	F (1)				experts conduct formal heuristic evaluation
Consistency Inspection	F (1)				expert checks consistency across products
Standards Inspection	F (1)				expert checks for standards compliance
Inquiry					
Contextual Inquiry	FI (1)				interviewer questions users in their env.
Field Observation	FI (1)				interviewer observes system use in user's env.
Focus Groups	FI (1)				multiple users participate in a disc. session
Interviews	FI (1)				one user participates in a disc. session
Surveys	FI (1)				interviewer asks user specific questions
Questionnaires	FI (1)	FI (1)			user provides answers to specific questions
Self-reporting Logs	FI (1)				user records UI operations
Screen Snapshots	FI (1)				user captures UI screens
User Feedback	FI (1)				user initiates comments
Analytical Modeling					
No Methods Surveyed					
Simulation					
Information Proc. Modeling			M (1)		mimic user interaction
Information Scent Modeling		M (1)			mimic Web site navigation
Automation Type					
Total	26	4	3	1	
Percent	76%	12%	9%	3%	

Table 1 Automation support for 57 Web UE methods. A number in parentheses indicates the number of UE methods surveyed for a particular method type and automation type. The effort level for each method is represented as: minimal (blank), formal (F), informal (I) and model (M). Two software tools provide automation support for multiple method types: Dome Tree visualization - log file analysis and information scent modeling; and WebVIP - performance measurement and remote testing.

Structural analysis of Web pages and sites

The Rating Game [Stein 1997] is an automated analysis tool that attempts to measure the quality of web pages using a set of easily measurable features. These aspects include: an information feature (word to link ratio), a graphics feature (number of graphics on a page), a gadgets feature (number of applets, controls, and scripts on a page), and so on. The tool reports these raw measures without providing guidance for interpreting them (with respect to a web page’s quality) or for improving the web page.

Two authoring tools from Middlesex University, HyperAT [Theng and Marsden 1998] and Gentler [Thimbleby 1997], perform a similar structural analysis at the site level. The goal of HyperAT is to support the creation of well-structured hyper-documents. Its structural analysis entails verifying that the breadths and depths within a page and at the site level fall within thresholds (e.g., depth less than three levels). Gentler [Thimbleby 1997] provides a similar structural analysis but focuses on the maintenance of existing sites rather than the design of new ones. The effectiveness of these structural analyses is questionable, because the thresholds have not been validated empirically.

The WebTango prototype [Ivory and Hearst 2002] uses predictive models to analyze the structure of individual pages and the site overall. Similarly to the Rating Game, the methodology entails computing 157 quantitative page-level and site-level measures. The measures assess many aspects of web interfaces, including the amount of text on a page, color usage, and consistency. These measures along with expert ratings from Internet professionals are used to derive statistical models of highly rated web interfaces. The models are then used in the automated analysis of new sites. For instance, the prototype reports predicted ratings along with a rationale for the predictions. The evaluator has to interpret this information to determine the most appropriate design changes.

WebRemUsine [Paternò and Paganelli 2001] provides an automated support to evaluate Web sites. It provides a certain number of measurements which are useful to detect some critical aspects of evaluated Web site. For example, the evaluator can see whether there are tasks particularly long, and if some of them exist, s/he can identify which pages create problems and whether they occur because of an excessive downloading time. The tool also displays the sequences of tasks performed and the associated sequences of Web pages accessed and provides lists of tasks performed (and pages accessed) ordered by duration or frequencies of performance (in case of tasks) or number of accesses (in case of pages). Patterns of tasks, actions and pages accessed are also automatically detected and displayed. The tool provides automatic support for analyzing whether the sequence of tasks performed is the optimal one for achieving the current goal and to count the number of useless actions performed and pages accessed. The tool is able to analyze not only single sessions but also groups of them, thus allowing the evaluator to understand whether a problem occurs often across many users or is only a problem limited to certain users and circumstances.

Analysis of HTML and CSS code

Several automated evaluation tools use HTML and CSS coding guidelines for web page assessment. The World Wide Web Consortium’s (W3C) "**HTML Validation Service1**" [W3C 2001] checks that HTML code conforms to W3C coding standards. Similarly, the W3C’s "**CSS Validation Service2**" [W3CC 2002] checks that style sheets conform to W3C standards. "**Weblint**" [Bowers 1996] and "**Dr.Watson**" [Addy & Associates 2000] also check HTML syntax and verify that links exists. In addition, "**Dr. Watson**" computes download speed.

Analysis of usability and accessibility

Many of the automated analysis and critique tools analyze the HTML code to ensure that it conforms to accessibility or usability guidelines—the Section 508 guidelines (US Federal standard) [CITA 2002], the W3C Content Accessibility guidelines [W3C 1999], or both. There is a lot of similarity between these two sets of guidelines, because the Section 508 guidelines are based on the W3C guidelines. (See [Thatcher 2002] for a side-by-side comparison.)

Overall, the tools address a broad range of user abilities [Ivory et al. 2003], but they focus on ensuring access, rather than improving user experiences. To address the limitations of these guidelines, Coyne and Nielsen [2001] proposed an additional 75 guidelines (e.g., minimize the use of graphics or small font sizes for text, avoid pop-up windows, separate browser windows, or cascading menus, and minimize the number of links on a page) derived from studies of users with vision and mobility impairments. These guidelines are embedded in the LIFT tool - Nielsen Norman Group Edition [UsableNet 2002a].

The tools are very similar in functionality. Essentially, the automated analysis tools—Automated Web site USability Analysis (AWUSA) [Tiedtke et al. 2002], Doctor HTML [Imageware Inc. 2001], Web Static Analyzer Tool (WebSAT) [Scholtz and Laskowski 1998], and WebTango [Ivory and Hearst 2002] — report guideline violations textually, graphically, numerically, or in some combination.

The critique tools— 508 Accessibility Suite3 [Macromedia 2001], AccessEnable [RetroAccess 2002], AccRepair/AccVerify [HiSoftware 2002a; HiSoftware 2002b], ALTifier [Vorburger 1999], A-Prompt [ATRC 2002], Bobby [WatchFire 2002], Dr. Watson [Addy & Associates 2000], InFocus [SSB 2002], LIFT [UsableNet 2000; UsableNet 2002a; UsableNet 2002b], PageScreamer [Crunchy Technologies, 2002], WAVE [PIAT 2001], and EvalWeb [Scapin et al. 2000]— also provide recommendations or assistance in correcting violations. Ivory et al. [2003] provide a detailed discussion of most of these tools.

The tools vary widely with respect to the type of assistance that they provide to evaluators when they attempt to correct violations. For example, the LIFT tools [UsableNet 2002a; UsableNet 2002b], 508 Accessibility Suite, AccVerify/ AccRepair, and A-Prompt provide wizards to walk evaluators through code modifications. The ALTifier and A-Prompt, suggest alternative text based on the page context and other heuristics. AccessEnable is a web-based tool that can make some changes to pages automatically; the evaluator can then download the modified pages. PageScreamer [Crunchy Technologies 2002] and InFocus [SSB 2002] are desktop applications that can also make changes.

Conforming to the guidelines embedded in these tools can potentially eliminate usability problems that arise due to poor HTML syntax (e.g., missing page elements) or guideline violations. Research on web site credibility [Fogg et al. 2000; Fogg et al. 2001; Kim and Fogg 1999] possibly suggests that some of the aspects assessed by these tools, such as broken links and other errors, may also affect usability due to the relationship between usability and credibility. However, Ratner et al. [1996] question the validity of HTML usability guidelines, since most have not been subjected to a rigorous development process as established guidelines for graphical interfaces. The authors' analysis of 21 HTML guidelines showed little consistency among them, with 75% of recommendations appearing in only one style guide. Furthermore, only 20% of HTML relevant recommendations from established graphical interface guidelines existed in the 21 HTML guidelines.

EvalWeb is one automated critique approach proposed to address this issue. It provides a framework for applying established graphical interface guidelines (e.g., the Smith and Mosier guidelines [Smith and Mosier 1986] and the Motif style guides [Open Software Foundation 1991]) to relevant HTML components. Even with EvalWeb, some problems, such as whether text will be understood by users, are difficult to detect automatically.

Visual analysis of Web pages

The existing approaches in this category are the Design Advisor approach [Faraday 2000] and the Page Layout Change approach [Farkas 2003]. The Design Advisor uses empirical results from eye tracking studies designed to assess the attentional effects of various elements, such as animation, images, and highlighting, in multimedia presentations [Faraday and Sutcliffe 1998]; these studies found motion, size, images, color, text style, and position to be scanned in this order. A preliminary eye-tracking study provides some support for the heuristics used in the web domain [Faraday 2001]. The Design Advisor determines and superimposes a scanning path on a web page, in which page elements are numbered to indicate the order in which users are likely to scan them. It currently does not provide suggestions for improving scanning paths.

As for the Page Layout Change approach [Farkas 2003], it analyzes the consistency of layouts across pages in a site. It entails decomposing each page on a user path into a visual hierarchy [Reichenberger et al. 1995], determining the type of changes (e.g., fusing of one or more regions, splitting of a region, or creation of a new region) that occur within the regions that comprise the visual hierarchy, and then computing a page layout score based on the changes. The author does not provide empirically validated thresholds for the page layout scores. Currently, an evaluator has to apply the methodology manually. We included it in our discussion due to its viability as a future automated analysis approach.

Textual analysis of Web page navigation

Cognitive Walkthrough for the Web (CWW) [Blackmon et al. 2002; Blackmon et al. 2003] is an automated critique approach that can identify potential navigation problems and provide guidance for correcting them. The approach entails the use of Latent Semantic Analysis[1] [Landauer and Dumais 1997] to contrast web page text (headings or links) to an evaluator-specified information-seeking goal (100–200 word narrative and correct link selection on the page). The LSA system computes several similarity measures for the two text inputs, based on the semantic space that the evaluator specifies (e.g., for grade nine reading ability). The authors provide LSA thresholds for three navigation problems: confusable heading or link text, unfamiliar heading or link text, and goal-specific competing heading or link text. Empirical studies have demonstrated that modifying page

[1] LSA is a statistical model of word usage that permits comparisons of semantic similarity between pieces of textual information [Foltz 1996].

text to fit within these thresholds improves usability. This approach is still under development.

2.3.6 Our Focus: UE by Guideline Review

As stated in the previous chapter, our central objective is to propose a methodology for automating Web usability and accessibility evaluation by static analysis of HTML code and by adopting the technique of guideline review. Thus, we will not go through another discussion or classification of UE methods, but we will focus our attention on one particular class (Inspection methods) and within this class, we will be interested in one method type (Guideline review). Thus, we will not consider methods using log file analysis, Surveys, GOMS analysis, etc. Interested reader can find an extensive survey of usability evaluation methods and tools in [Ivory 2003].

This limitation is motivated by the following raisons:

- As we saw above, there are many extensive surveys about UE methods, and we don’t have the intention to "reinvent the wheel".
- The methodology that we propose in this work concerns automation of only this type of UE, so we will focus our attention on identifying major shortcomings of UE tools adopting this type of methods in order to propose solutions for them.
- UE by guideline review provides the fullest automation support [Farenc and Palanque 1999] because it enables evaluation without requiring completion of freely chosen tasks (informal use) or specially selected tasks (formal use) [Ivory and Hearst 2001].
- Tools for UE by guideline review are gaining wide acceptance among Web designers and evaluators, and we aim to propose a methodology that can have rapid practical application. For example, the W3C’s guidelines on evaluating Web sites for accessibility [W3C 2001a] recommend that designers use two of three automated evaluation tools by guideline review (Bobby [WatchFire 2002]; WAVE [Pennsylvania’s Initiative on Assistive Technology 2001]; A-Prompt [ATRC 2003]) to identify and correct potential accessibility problems during the design process.

Guidelines

We hereby define a *Web usability guideline* as any statement ensuring some adequacy of a particular user interface of a Web site with respect to a particular context of use where a given user population has to fulfill interactive tasks with a given system [Scapin et al. 2000].

Uses

Generally speaking, guidelines are mostly used during [Vanderdonckt 1999]:

- **The specification phase (Discovery)**: a set of guidelines is delimited as requirements for the future UI;
- **The design phase (Design Exploration)**: guidelines are exploited in order to decide an appropriate value for each design option by considering the context;

- **The prototyping phase (Design Refinement)**: guidelines are exploited to obtain as soon as possible a static or working UI prototype that can be showed, tested and evaluated;
- **The programming phase (Production)**: guidelines are gathered to guide, orient, decide, ensure a UI development;
- **The evaluation phase (Quality Assurance)**: the resulting UI is evaluated with respect to guidelines that are often ones that have been selected in previous phases;
- **The documentation and certification phase (Quality Assurance)**: guidelines which have been manipulated in previous phases are instructed for documenting an interactive application for communication, reuse, maintenance or commercial promotion purposes.

Sources

Many guidelines have been published in document sources throughout the literature. These sources fall into five categories [Bastien and Scapin, 1995; Vanderdonckt 1999]:

- **Design rules**: they comprise a set of functional and/ or operational specifications that specify the design of a particular user interface. These specifications are presented in a form that requires no further interpretation, either from designers or from developers. Their straightforward format allows an immediate exploitation (ex. "*Each page should contain a KEYWORDS meta-tag*").
- **Compilations of guidelines**: they comprise several prescriptions written for a wide range of user interfaces. Each prescription is presented as a statement, sometimes along with examples, with or without clarifying explanations and comments. Each prescription generally results from a human consensus among guideline users. This consensus is less relevant once a prescription is experimentally tested and verified. The scope of the compilations can range from a small set of guidelines dedicated to a particular usability feature (e.g., an interaction technique) to an extensive collection of guidelines covering a family of tasks and domains. Such guidelines are appropriate for multimedia applications, for interactive kiosks, to ensure accessibility [Vanderheiden et al. 1997], to design web site in general [Detweiler and Omanson 1996]. Some guidelines are validated by experimental results provided by user testing, laboratory experiences or other techniques, while others are not.
- **Style guides**: they comprise a set of guidelines and/or functional or non-functional specifications aiming at consistency for a collection of distinct user interfaces. This collection can be based on an operating system, on a software editor, by a particular physical environment such as SUN [Levine 1996], by a domain of human activity or by a corporate. In particular, Ratner et al. [1996] identified that only 20% of recommendations from established style guides that apply to the web were found in web style guides. They also found out that little consistency existed among the 21 considered web style guides, with 75% of recommendations appearing in only one style guide.
- **Standards**: they comprise a set of functional and/or operational specifications intended to standardize design. Standards are promulgated by national or international organizations for standardization. They can be military,

governmental, civil or industrial. There is no web dedicated standard we are aware of, but some important general standards are good candidates to tailor their guidelines to suit the exact issues faced by web sites, e.g., ISO 9241 [ISO, 1999], HFES/ANSI 200 [HFES 1997].

- **Ergonomic algorithms**: such algorithms typically aggregate single design rules into a comprehensive and systematic procedure that can be applied more quickly than a series of single guidelines. In this way, they introduce more flexibility by enabling designers to select parameters required for design rules and they prevent designers to inadvertently forget some design rules. They more usually appear as a software component that implements an algorithm rather than a paper procedure. Such algorithms are primarily intended to design a series of web pages that automatically respect guidelines by construction.

Although the five types of sources contain multiple types of guidelines, they often contain guidelines requiring some interpretation at a certain level of abstraction [Scapin et al. 2000].

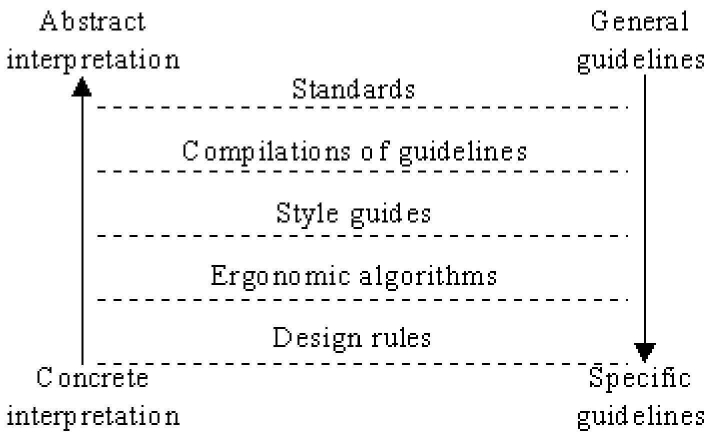


Figure 2.3: Location of sources containing guidelines.

Web Guidelines

Contrarily to the domain of usability for graphical user interfaces, Web usability guidelines in these categories are often produced by consensus, common sense or observation of best practices. Guidelines validated by experimental results are rare and hard to find out. Web usability guidelines tend to address Web specific usability issues, but not necessarily usability issues for graphical user interfaces adapted or specialized for the Web. However, we can note that “usability guidelines that are valid for graphical user interfaces must also be considered as potential usability guidelines for the Web, unless some contradiction or invalidation is appearing” [Scapin et al. 2000].

Also for accessibility issues, guidelines have been proposed [WAI 1999]. "Web Content Accessibility Guidelines 1.0", published by the Web Accessibility Initiative [WAI 1999], are intended for all Web content developers (page authors and site designers) and for developers of authoring tools.

Even if guidelines are useful for automated inspection, they are insufficient. They should not be considered in isolation: often guidelines need to be supplemented by a suitable method and a clear process that leads them to unambiguous interpretation.

Although guidelines can be used in conjunction with guidelines checklists, standard inspection, and consistency inspection, a variant of heuristic evaluation has been set up where the usual set of 10 heuristics [Nielsen & Mack 1994] is replaced by a set of specific guidelines. This replacement came from the observation of heuristics which, though cost-effective, have been considered too general and not expressive enough to be interpreted and applied effectively. Conversely, lists of guidelines are potentially long. If they are not specific, a lot of time to interpret them is likely to be devoted since the whole user interface and its components should be checked against them. However, if developing a tool for automated or computer-aided evaluation of these guidelines is the ultimate goal, then such shortcomings are important and hard to solve. If providing people with assistance and guidance in applying and evaluating guidelines is the ultimate goal, then shortcomings are less important [Scapin et al. 2000].

2.4 Overview of automated tools for evaluation by Guideline Review

2.4.1 Tools for traditional UI

In fact, evaluation of user interfaces by guideline review started long time ago, even before the Web. Several quantitative measures have been proposed for evaluating traditional interfaces. Tullis [1983] derived size measures (Overall Density, Local Density, Number of Groups, Size of Groups, Number of Items, and Layout Complexity). [Streveler and Wasserman 1984] proposed "boxing" "hotspot" and "alignment" analysis techniques. These early techniques were designed for alphanumeric displays, while more recent techniques evaluate WIMP interfaces. Vanderdonckt and Gillo [1994] proposed five visual techniques (Physical Composition, Association and Dissociation, Ordering, and Photographic Techniques), which identified more visual design properties than traditional balance, symmetry and alignment measures. Rauterberg [1996] proposed and validated four measures (Functional Feedback, Interactive Directness, Application Flexibility, and Dialog Flexibility) to evaluate WIMP UIs. Quantitative measures have been incorporated into automated layout tools [Bodart et al. 1994] as well as several automated analysis tools [Mahajan and Shneiderman 1997; Parush et al. 1998; Sears 1995].

Parush et al. [1998] developed and validated a tool for computing the complexity of dialog boxes implemented with Microsoft Visual Basic. The tool considers changes in the size of screen elements, the alignment and grouping of elements, as well as the utilization of screen space in its calculations. Usability studies demonstrated that tool results can be used to decrease screen search time and ultimately to improve screen layout. AIDE (semi-Automated Interface Designer and Evaluator) [Sears 1995] is a more advanced tool that helps designers assess and compare different design options using quantitative task-sensitive and task-independent metrics, including efficiency (i.e., distance of cursor movement), vertical and horizontal alignment of elements, horizontal and vertical balance, and designer-specified constraints (e.g., position of elements). AIDE also employs an optimization algorithm to automatically generate initial UI layouts. Studies with

AIDE showed it to provide valuable support for analyzing the efficiency of a UI and incorporating task information into designs.

Sherlock [Mahajan and Shneiderman 1997] is another automated analysis tool for Windows interfaces. Rather than assessing ergonomic factors, it focuses on task-independent consistency checking (e.g., same widget placement and labels) within the UI or across multiple UIs; user studies have shown a 10-25% speedup for consistent interfaces [Mahajan and Shneiderman 1997]. Sherlock evaluates visual properties of dialog boxes, terminology (e.g., identify confusing terms and check spelling), as well as button sizes and labels. Sherlock evaluates any Windows UI that has been translated into a special canonical format file; this file contains GUI object descriptions. Currently, there are translators for Microsoft Visual Basic and Microsoft Visual C++ resource files.

The KRI/AG tool (Knowledge-based Review of user Interface) [Lowgren and Nordqvist 1992] is an automated system that checks the guideline conformance of X Window interface designs. It contains a knowledge base of guidelines and style guides, including the Smith and Mosier guidelines [Smith and Mosier 1986] and the Motif style guides [Open Software Foundation 1991]. It uses this information to automatically critique a UI design and generate comments about possible problems in the design. IDA (user Interface Design Assistance) [Reiterer 1994] also embeds rule-based (i.e., expert system) guideline checks within a UIMS.

SYNOP [Kolski 1993] is a similar automated system that performs a rule-based critique of a control system application. SYNOP also modifies the UI model based on its evaluation.

CHIMES (Computer-Human Interaction ModElS) [Jiang et al. 1993] assesses the degree to which NASA's space-related critical and high risk interfaces meet human factors standards.

Unlike KRI/AG, IDA, SYNOP, and CHIMES, Ergoval [Farenc and Palanque 1999] is widely applicable to WIMP UIs on Windows platforms. It organizes guidelines into an object based framework (i.e., guidelines that are relevant to each graphical object) to bridge the gap between the developer's view of an interface and how guidelines are traditionally presented (i.e., checklists). This approach is being incorporated into a Petri net environment [Palanque et al. 1999] to enable guideline checks throughout the development process.

All WIMP approaches are highly effective at checking for guidelines that can be operationalized. These include computing quantitative measures (e.g., the size of screen elements, screen space usage, and efficiency) and checking consistency (e.g., same widget size and placement across screens). All of the tools have also been empirically validated. However, the tools cannot assess UI aspects that cannot be operationalized, such as whether the labels used on elements will be understood by users. [Farenc et al. 1996] show that only 78% of a set of established ergonomic guidelines could be operationalized in the best case scenario and only 44% in the worst case.

All methods also suffer from limited applicability (interfaces developed with Microsoft Visual Basic or Microsoft Visual C). The tools appear to be straight forward to learn and use, provided the UI is developed in the appropriate environment. Another drawback of approaches that are not embedded within a UIMS (e.g., SYNOP) is that they require considerable modeling and learning effort on the part of the evaluator. All methods, except Ergoval, also suffer from limited applicability [Ivory 2001].

2.4.2 Web tools

Contrarily to WIMP UI, Web UI use very limited number of languages (mainly HTML), which makes it easier to develop widely applicable evaluation tools. [Ivory 2003] elaborated a very detailed survey and analysis for existing automated Web evaluation methods and tools. We think that we can not do better than her because:

- She explored a big number of existing tools during a long period of time.
- We don't have access to many of these tools because they are available commercially only.
- She enforced her theoretical study with empirical study of some of these tools.

Ivory selected the following guideline review tools:

- 508 Accessibility Suite [Macromedia 2001]
- A-Prompt [ATRC 2002]
- AccessEnable [RetroAccess 2002]
- AccVerify/AccRepair [HiSoftware 2002b; HiSoftware 2002a]
- Bobby [WatchFire 2002]
- Doctor HTML [Imageware Inc. 2001]
- LIFT-NNG [UsableNet 2002a]
- W3C HTML Validator [W3C 2001c]
- WAVE [PIAT 2001]
- WebTango [Ivory and Hearst 2002]

With the exception of the WebTango prototype, all tools were either available commercially or commercially viable (i.e., robust applications).

Ivory initially ran the tools on the same web page in the fall of 2002 and then updated the assessments in May of 2003. She analyzed the reports generated by each tool to determine the total number of errors, number of unique errors, and the number of false errors (i.e., an issue was inaccurately identified as an error). Some tools report alerts for issues that are relevant in general, but not detected on the page; we aggregate the total and unique number of alerts with the total and unique number of errors. Ivory coded the guideline violations so that she could compare the reported errors across tools.

We begin this section with a characterization of the user abilities (e.g., use of keyboard and mouse devices and visual, cognitive, and hearing abilities) that designers need to consider when designing web sites. Section 2.5.2 contrasts guideline review tools with respect to the user abilities that they support in their assessments.

We conclude with a comparison of evaluation results for above guideline review tools; we summarize Ivory's findings from running the tools on a single web page. Although the comparisons show the range of support provided by existing tools, they also demonstrate that there is very little consistency in evaluations across tools. Nonetheless, Ivory's analyses can help practitioners to identify and employ tools that are most appropriate for their web site design contexts.

2.4.3 Characterization of user abilities

The Web is intended to enable broader access to information and services than was previously available. However, web site usability and accessibility continues to be a pressing problem. Although numerous assistive devices, such as screen readers and special keyboards, facilitate web site use, these strategies may not improve a user's ability to find information, purchase products, and complete other tasks on sites. For example, sites may not have links to help those who use screen readers to skip over navigation bars, or sites may not enable users to increase the font size of text so that they can read it.

Understanding diverse user abilities is important so that we can better support them in web site designs. We use the following categories to characterize user abilities; a detailed discussion can be found in [W3C 2001b].

- **Motor abilities**: we decompose this category into:
    - *Mouse Use*: Users may be able to use a mouse or they may have partial (limited) or no ability to use one. Mouse use may be reduced for physiological reasons. For instance, a motor impairment or blindness may make it difficult to use a mouse.
    - *Keyboard Use*: Similarly to mouse use, users may (fully or partially) or may not be able to use a keyboard. We separate keyboard and mouse use because computers are controlled almost exclusively by these two devices, they are required by different aspects of web browsing interfaces, and they depend on motor skills in different ways.
- **Vision abilities**: Users may not have visual impairments that impede their ability to view web pages visually. However, some users have limited visual ability (due to color blindness, low vision, or other impairments) or no visual ability.
- **Hearing abilities**: Similarly to visual abilities, users may (fully or partially) or may not be able to hear sounds on web pages.
- **Cognition abilities**: We consider two aspects: cognition attention and comprehension. Users may be able to fully or partially attend to web pages. For instance, attention deficits may impede users' attention. Similarly, users may be able to fully or partially comprehend information on web pages; learning and memory impairments may impede users' comprehension.

These diverse user abilities need to be accommodated within web site designs, if possible. We discuss design-level support approaches in the next section.

2.4.4 Evaluating whether Web sites support diverse user abilities

Most guideline review tools check whether sites conform to the Section 508 guidelines, the W3C Content Accessibility guidelines [W3C 1999], or both. There is a lot of similarity between these two sets of guidelines, because the Section 508 guidelines are based on the W3C guidelines. Overall, the tools evaluate whether most user abilities (discussed in Section 1) are supported in web site designs, but they emphasize enabling access, rather than improving user experiences.

To address the limitations of W3C and Section 508 guidelines, Coyne and Nielsen [2001] proposed an additional 75 guidelines derived from studies of users with vision and mobility impairments. Example guidelines include: minimize the use of graphics or small font sizes for text, avoid pop-up windows, separate browser windows, or cascading menus, and minimize the number of links on a page. These guidelines are embedded in the LIFT tool - Nielsen Norman Group Edition (LIFT NNG) [UsableNet 2002a].

Table 2.2 provides a synopsis of the range of user abilities that existing guideline review tools support [Ivory 2003]. It only characterizes the diminished or limited user abilities; Appendix A provides a summary for all the tools listed in the table.

Depending on the intended site users, the table can help designers to determine the evaluation tools that are most appropriate to use. Overall, many tools address limited visual abilities, a few address other non-visual impairments, and no tools address limited keyboard use. Bobby [WatchFire 2002], LIFT-NNG [UsableNet 2002a], WebSAT [NIST 2001], and WAVE [PIAT 2001] provide broader coverage of user abilities than all the other tools. We describe the types of evaluations that the tools conduct and the limitations of their assessments in the remainder of this section.

Motor abilities

Mouse Use

A few tools check to see if web pages enable alternative forms of input (e.g., keyboard or Braille device use) to address the needs of users with some or no ability to use a mouse. For example, Bobby [WatchFire 2002] and WAVE [PIAT 2001] look for keyboard shortcuts for forms and links or for equivalent ways to handle events like mouse roll-overs. However, the tools do not analyze the shortcuts to see if they are meaningful, consistent with common interface shortcuts, or conflict with each other (e.g., the same shortcut may be specified for more than one action). Bobby and WAVE also check for an explicit tabbing order on pages, but there is no analysis of the tabbing order, for instance to see if it is logical. LIFT-NNG [UsableNet 2002a] checks for the use of cascading menus, which the developers determined to be problematic for users with disabilities [Coyne and Nielsen 2001].

Keyboard Use

To accommodate limited keyboard use, the guideline review tools could check to see if web pages enable speech input, rather than physical input (keyboard, Braille device, or mouse input). Although the tools check for the ability to output speech (e.g., checks for the presence of alternative text for images, labels for form elements, and so on), they currently do not assess the ability for users to employ speech-based navigation, wherein they send voice commands to control browsers [Christian et al. 2000]. Additionally, these tools do not consider the needs of users for whom keyboard input is limited or slow. For example, a text entry area for entering a zip code could be simplified by replacing it with 5 menus each containing the digits 0 through 9. Note that a user with normal keyboard use may find the series of menus more cumbersome than simply typing the zip code. This difference in user preferences shows that authoring-time solutions may not always be appropriate, and thus that automated transformation tools are a necessary component of true universal access [Ivory et al. 2003].

Table 2.2: Diminished user abilities addressed by guideline review tools.[a]

	Mouse Use		*Keyboard Use*		*Vision*		*Hearing*		*Cognition*	
Tool	P	N	P	N	P[b]	N	P	N	PA	PC[c]
508 Suite						√				
A-Prompt						√				
AccessEnable						√	√	√		
AccVerify/AccRepair						√				
ALTifier						√				
Bobby	√	√			√	√			√	√
Doctor HTML						√				√
LIFT (all versions)	√	√								
LIFT-NNG	√	√			√	√				√
PageScreamer						√				
W3C HTML Validator					√					
WAVE	√	√				√			√	
Weblint						√				
WebSAT					√	√			√	√
WebTango					√				√	√

[a] P columns denote partial ability. N columns denote no ability. The PA and PC columns denote partial attention and partial comprehension, respectively.
[b] Partial vision includes low vision and color blindness.
[c] Partial comprehension includes learning and memory impairments.

Vision abilities

Nearly all guidelines embedded in the existing tools address the needs of users with little or no vision. Most checks entail looking for the existence of page and frame titles, alternative text for images and other objects, headings for tables, and labels for form elements so that screen readers can process this information [ATRC 2002; Bowers 1996; Crunchy Technologies 2002; NIST 2001; PIAT 2001; RetroAccess 2002; UsableNet 2002a; Vorburger 1999; WatchFire 2002]. Several tools provide wizards to help developers to correct such violations (e.g., [ATRC 2002; HiSoftware 2002a; Macromedia 2001; UsableNet 2002a]), and other tools, such as the ALTifier [Vorburger 1999] and A-Prompt [ATRC 2002], suggest alternative text based on the page context and other heuristics. LIFT-NNG also checks for the presence of table summaries. None of the tools, however, can assess whether alternative text, titles, labels, headings, or table summaries are meaningful.

Bobby, AccessEnable, and LIFT-NNG check whether pages enable users to skip repetitive navigation links. However, the tools do not check whether skipping links actually take the user to the appropriate areas on web pages. LIFT-NNG also checks to see if the page launches separate browser windows, which can impede navigation.

AccessEnable, Doctor HTML, A-Prompt, and WAVE conduct rudimentary analyses of table and form structures. For example, A-Prompt depicts the linear reading order for table contents, and WAVE depicts the linear reading order for the entire page. Developers have to manually evaluate the reading order to determine if it is logical or efficient; automated tools do not support this type of analysis.

The tools perform several checks to address the needs of users with partial vision, such as analyzing color usage [Ivory and Hearst 2002a; NIST 2001], font and image sizes [Ivory and Hearst 2002a; UsableNet 2002a], and checking for the separation of content from presentation via style sheets [Ivory and Hearst 2002a; RetroAccess 2002; WatchFire 2002; W3C 2002]. For example, the W3C CSS Validation Service [W3C 2002] enables developers to see if their style sheets conform to W3C CSS coding standards. As another example, Bobby checks to see if pages use relative positioning (i.e., widths and heights expressed as percentages to facilitate screen magnification) rather than absolute positioning (i.e., widths and heights expressed as pixels).

The WebTango prototype [Ivory and Hearst 2002] analyzes the quality of text and background color combinations, using Murch's [1987] study of color combination preferences as a basis; it reports the number of good, neutral, and poor color combinations along with the specific combinations used. It also reports on the sizes used for images and text; these sizes are compared to statistical models derived from an analysis of over 5300 web pages [Ivory and Hearst 2002]. Similarly, it reports ranges for the number of links, number of words within each text link, and the number of graphics. It also reports on the use of text and link clustering and text alignment, which may be helpful to users with partial vision. The WebTango prototype is a rudimentary research system, and is not currently robust enough for wide use.

Hearing abilities

Both the W3C and Section 508 guidelines require text equivalents for audio, video, or other multimedia formats. In addition, they require synchronization of these text equivalents with their sources. Most of the tools check for alternative text for objects, which screen readers can read to a vision-impaired user (see preceding discussion). However, there are no checks to see if this alternative text is easily available to a hearing-impaired user. Furthermore, the tools cannot check whether text equivalents or alternative text are consistent with their sources.

AccessEnable seems to be the only tool that can flag whether there is some synchronization between the text equivalents and their sources, though it is not clear how this assessment is accomplished, because synchronization is difficult to check automatically.

Cognition abilities

W3C and Section 508 guidelines address animation, which can cause problems for users with attention deficits. Bobby, WebSAT, and the 508 Accessibility Suite check for animated images or scrolling text. The WebTango prototype also quantifies the use of animated images and provides ranges based on an analysis of over 5300 web pages. None of the tools can assess whether there are ways stop animations.

To address the needs of users with comprehension impairments, the guidelines suggest that pages should be well-structured and consistent within a site. For example, Bobby, the LIFT tools, and others check that pages and frames have titles, which can help to orient users. WAVE checks for the use of structural tags, such as headings and lists, to determine if they are used properly.

As another example, the WebTango prototype quantifies and provides ranges for the amount of text, links, images, and other elements, as a way to address potential cognitive overload. It also assesses download speed, consistency, the use of link and text clustering, and reading complexity, all of which may impede web site use by users with comprehension impairments. Similarly, Doctor HTML checks for spelling errors and analyzes images (e.g., image size and number of colors). WebSAT also reports the total size of graphics. LIFT-NNG, WebSAT, and the WebTango prototype evaluate other text, link, and image formatting that may impede web site use. For example, WebSAT reports whether web pages use non-default link colors, less than 2–5 words in text links, line breaks before or after links, horizontal rules, or separate browser windows. WebSAT also reports a ratio of links to words, which may provide some indication of text density. LIFT-NNG reports on the use of small text, too short text links or button labels (less than six characters), too many outgoing links (more than twenty) or images, long cascading menus, separate browser windows, and inadequate space between vertically aligned text links.

Guideline review tools assess many aspects that could have negative effects on comprehension, but the key aspect that is missing is an analysis of the content itself. None of the tools provide concrete insight about the content's quality, cohesion, appropriateness, readability, and so on. Although the tools may be able to improve the presentation of good content, they cannot improve on poor content.

2.4.5 Comparison of tools Assessments

The preceding section compared the diminished user abilities that are considered in assessments by existing guideline review tools. Although multiple tools may address a specific impairment, there may be differences in how they assess support for the impairment.

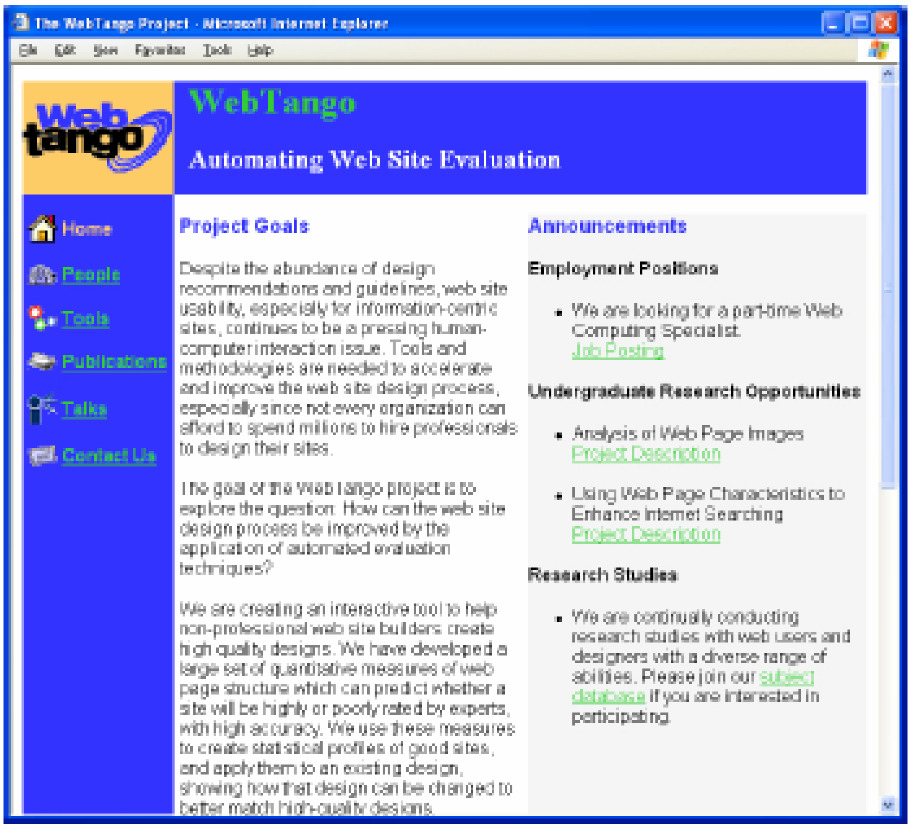
To better understand these differences, Ivory [2003] used the ten guideline review tools to evaluate a single web page (figure 2.4). Specifically, she examined the consistency of assessments across tools. In addition, she characterized the strengths and weaknesses of the tools with respect to their ease of use, their evaluation accuracy, and the usefulness of their reported results. We will summarize her results in this section.

According to Ivory [2003]:

Ease of Use

The tools were straightforward to use, at least for the evaluations. A-Prompt was somewhat challenging to use, because it requires the practitioner to specify a file directory and then characterize each table as a layout or data table, before assessments can be completed. Most reports were also straightforward to read, but there were some exceptions, for instance the WAVE report (Figure 2.5). The WAVE tool annotates web pages with icons to indicate guideline violations. There are numerous unintuitive icons, and we had to repeatedly consult the documentation to determine the meaning of each violation. WAVE also indicates the linear reading order on web pages; we initially confused these indicators with the guideline violation indicators.

Ivory found it difficult to explore results or to make repairs with A-Prompt, LIFT-NNG, and the 508 Accessibility Suite. Each tool presents a list of terse violations within a small window (see the window in the middle of Figure 2.7); the LIFT-NNG presents the same guideline violations repeatedly (one for each occurrence).


The WebTango Project - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Web tango
Home
People
Tools
Publications
Talks
Contact Us

WebTango

Automating Web Site Evaluation

Project Goals

Despite the abundance of design recommendations and guidelines, web site usability, especially for information-centric sites, continues to be a pressing human-computer interaction issue. Tools and methodologies are needed to accelerate and improve the web site design process, especially since not every organization can afford to spend millions to hire professionals to design their sites.

The goal of the WebTango project is to explore the question: How can the web site design process be improved by the application of automated evaluation techniques?

We are creating an interactive tool to help non-professional web site builders create high quality designs. We have developed a large set of quantitative measures of web page structure which can predict whether a site will be highly or poorly rated by experts, with high accuracy. We use these measures to create statistical profiles of good sites, and apply them to an existing design, showing how that design can be changed to better match high-quality designs.

Announcements

Employment Positions

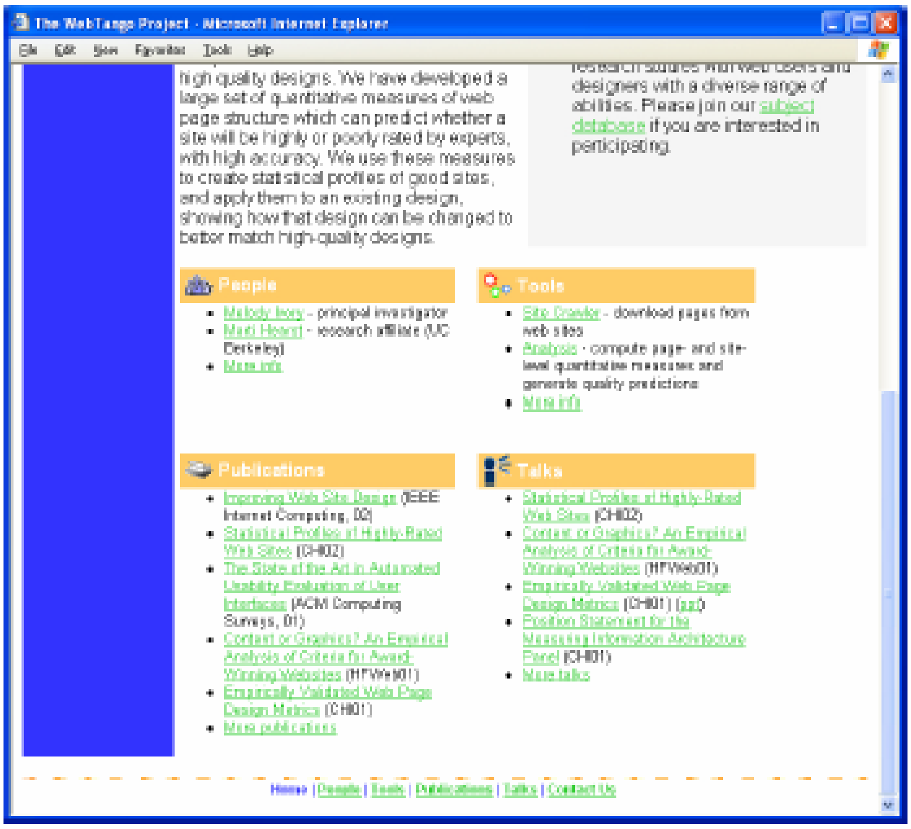
- We are looking for a part-time Web Computing Specialist.
Job Posting

Undergraduate Research Opportunities

- Analysis of Web Page Images
Project Description
- Using Web Page Characteristics to Enhance Internet Searching
Project Description

Research Studies

- We are continually conducting research studies with web users and designers with a diverse range of abilities. Please join our subject database if you are interested in participating.


The WebTango Project - Microsoft Internet Explorer
File Edit View Favorites Tools Help

high quality designs. We have developed a large set of quantitative measures of web page structure which can predict whether a site will be highly or poorly rated by experts, with high accuracy. We use these measures to create statistical profiles of good sites, and apply them to an existing design, showing how that design can be changed to better match high-quality designs.

research studies with web users and designers with a diverse range of abilities. Please join our subject database if you are interested in participating.

People

- Melody Ivory - principal investigator
- Marti Hearst - research affiliate (UC Berkeley)
- More info

Tools

- Site Crawler - download pages from web sites
- Analysis - compute page- and site-level quantitative measures and generate quality predictions
- More info

Publications

- Improving Web Site Design (IEEE Internet Computing, 02)
- Statistical Profiles of Highly-Rated Web Sites (CHI02)
- The State of the Art in Automated Usability Evaluation of User Interfaces (ACM Computing Surveys, 01)
- Content or Graphics? An Empirical Analysis of Criteria for Award-Winning Websites (HFWeb01)
- Empirically Validated Web Page Design Metrics (CHI01)
- More publications

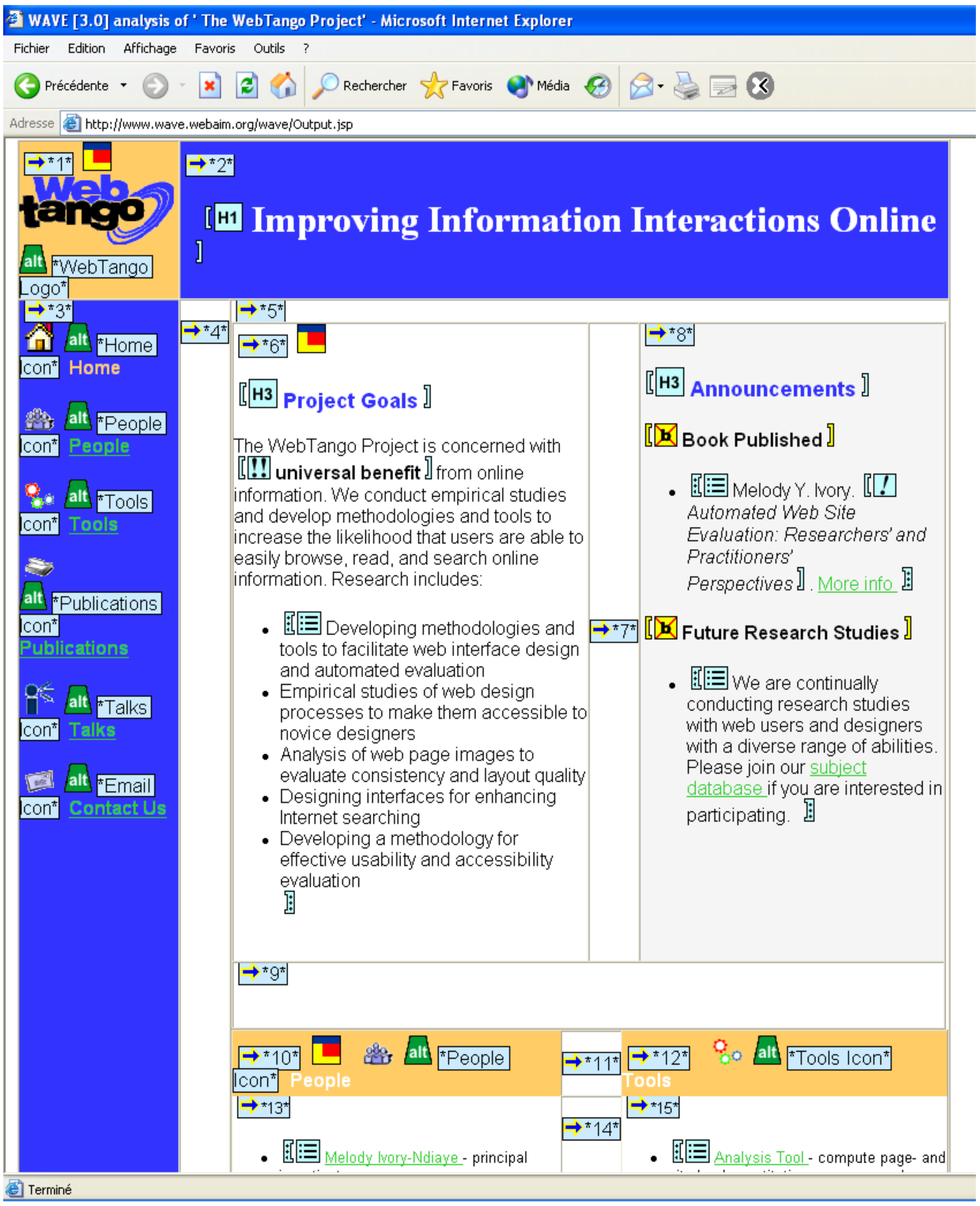
Talks

- Statistical Profiles of Highly-Rated Web Sites (CHI02)
- Content or Graphics? An Empirical Analysis of Criteria for Award-Winning Websites (HFWeb01)
- Empirically Validated Web Page Design Metrics (CHI01) (ppt)
- Position Statement for the Measuring Information Architecture Panel (CHI01)
- More talks

Home | People | Tools | Publications | Talks | Contact Us

Figure 2.4: Web page used for the comparison of guideline review tools. The page covers two vertical screens and does not conform to all accessibility and usability guidelines. The page uses tables and style sheets to control layout and formatting; it does not use scripts, applets, animated images, or other implementation technology.

After selecting an error from the list, the tools provide additional details and repair functionality. The repair functionality is accessed from yet another window. Figure 2.6 depicts the cumbersome task flow for evaluating and repairing a web page using the LIFT-NNG tool from within Macromedia Dreamweaver.


WAVE [3.0] analysis of ' The WebTango Project' - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

Précédente Rechercher Favoris Média

Adresse http://www.wave.webaim.org/wave/Output.jsp

1 alt *WebTango Logo*

2 H1 Improving Information Interactions Online

3 alt *Home Icon* Home

alt *People Icon* People

alt *Tools Icon* Tools

alt *Publications Icon* Publications

alt *Talks Icon* Talks

alt *Email Icon* Contact Us

4 *5* *6*

H3 Project Goals

The WebTango Project is concerned with universal benefit from online information. We conduct empirical studies and develop methodologies and tools to increase the likelihood that users are able to easily browse, read, and search online information. Research includes:

- Developing methodologies and tools to facilitate web interface design and automated evaluation
- Empirical studies of web design processes to make them accessible to novice designers
- Analysis of web page images to evaluate consistency and layout quality
- Designing interfaces for enhancing Internet searching
- Developing a methodology for effective usability and accessibility evaluation

7 *8*

H3 Announcements

Book Published

- Melody Y. Ivory. *Automated Web Site Evaluation: Researchers' and Practitioners' Perspectives*. More info

Future Research Studies

- We are continually conducting research studies with web users and designers with a diverse range of abilities. Please join our subject database if you are interested in participating.

9

10 alt *People Icon* People *11* *12* alt *Tools Icon* Tools

13 *14* *15*

- Melody Ivory-Ndiaye - principal
- Analysis Tool - compute page- and

Terminé

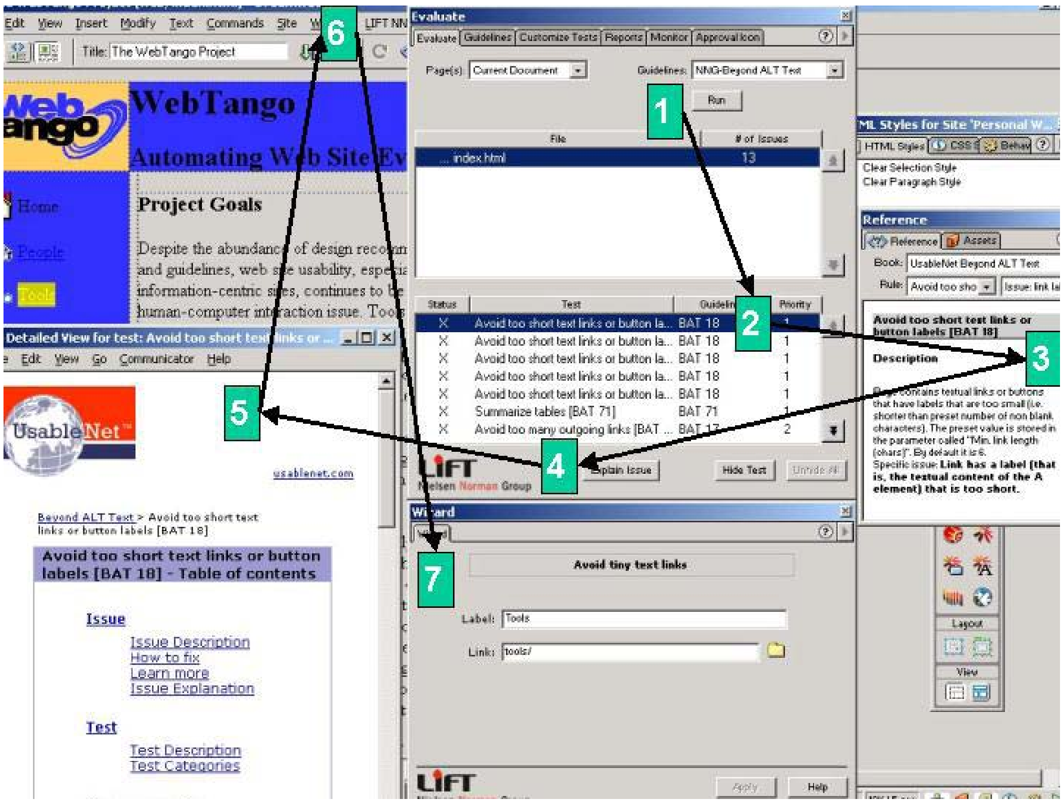
Figure 2.5: Evaluation report produced by WAVE. WAVE overlays guideline violations on the actual web page. It uses numerous icons to flag potential issues and also depicts the page reading order (arrows with numbers).

Errors reported across tools

Analysis of the reported guideline violations revealed that they represent 45 unique errors (e.g., avoid small text, identify the language of the text, and group related elements when possible). 35 of the violations can be linked to W3C guidelines and 9 violations to Section508 guidelines; only 7 violations linked to both a W3C and Section 508 guidelines. The web site design was analyzed to determine whether each violation was an accurate assessment. Only 21 violations (47 percent) seemed to be valid. Inaccurate assessments were reported for issues that could not be evaluated automatically or were not automated by the tools; example violations include: “avoid using shrunk-down pictures of other pages”, “do not use the same link phrase more than once when the links point to different URLs”, and “flicker is distracting and may be dangerous to some users”.

Figure 2.7 shows that most of the guideline violations relate to vision, coding, cognition, and general accessibility and usability issues; the latter category comprises guidelines that are not targeted at a specific user ability, rather they relate to usability and accessibility, in general.

Figure 2.8 shows that most guideline violations are considered to be priority two (i.e., moderate [World Wide Web Consortium, 2001c]); violations include: “avoid deprecated features of W3C technologies” (coding), “avoid using shrunk-down pictures of other pages” (general usability), and “check that the foreground and background colors contrast sufficiently with each other” (vision).



Page(s): Current Document   Guidelines: NNG-Beyond ALT Text

File	# of Issues
index.html	13

Status	Test	Guideline	Priority
X	Avoid too short text links or button la...	BAT 18	1
X	Avoid too short text links or button la...	BAT 18	1
X	Avoid too short text links or button la...	BAT 18	1
X	Avoid too short text links or button la...	BAT 18	1
X	Avoid too short text links or button la...	BAT 18	1
X	Summarize tables [BAT 71]	BAT 71	1
X	Avoid too many outgoing links [BAT ...	BAT 17	2

Book: UsableNet Beyond ALT Text
Rule: Avoid too sho   Issue: link la

Avoid too short text links or button labels [BAT 18]

Description

Page contains textual links or buttons that have labels that are too small (i.e. shorter than preset number of non blank characters). The preset value is stored in the parameter called "Min. link length (chars)". By default it is 6.
Specific issue: **Link has a label (that is, the textual content of the A element) that is too short.**

Beyond ALT Text > Avoid too short text links or button labels [BAT 18]

Avoid too short text links or button labels [BAT 18] - Table of contents

Issue
- Issue Description
- How to fix
- Learn more
- Issue Explanation

Test
- Test Description
- Test Categories

Avoid tiny text links
Label: Tools
Link: tools/

Figure 2.6: Screenshot of the LIFT-NNG tool within Macromedia Dreamweaver. The numbers depict the task flow for a practitioner who wants to evaluate a page (step 1), explore the report (steps 2–3), explore extra guideline information (steps 4–5), and then use the repair wizard to make appropriate modifications (steps 6–7).

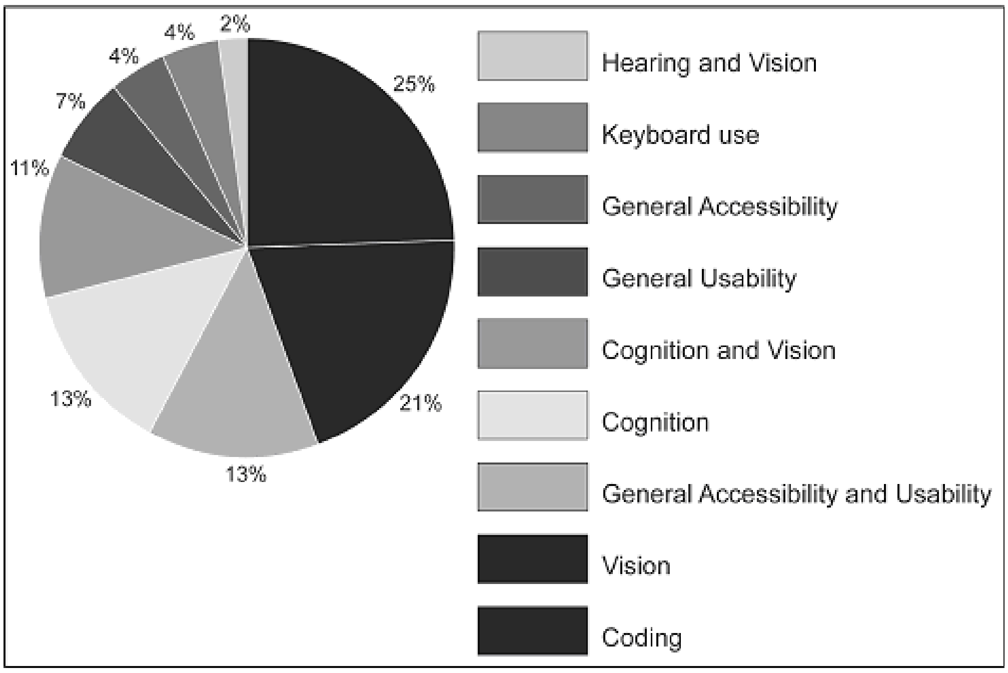


Figure 2.7: Aspects evaluated across tools. The pie chart shows the percent represented by each category in counter clockwise order from the top left. We associated the 45 guidelines with the user abilities summarized in Table 11.1, if possible. In some cases, a guideline referred to multiple abilities, which we reflect in the graph. We do not distinguish the degrees (e.g., full or partial) of user abilities that the tools assessed.

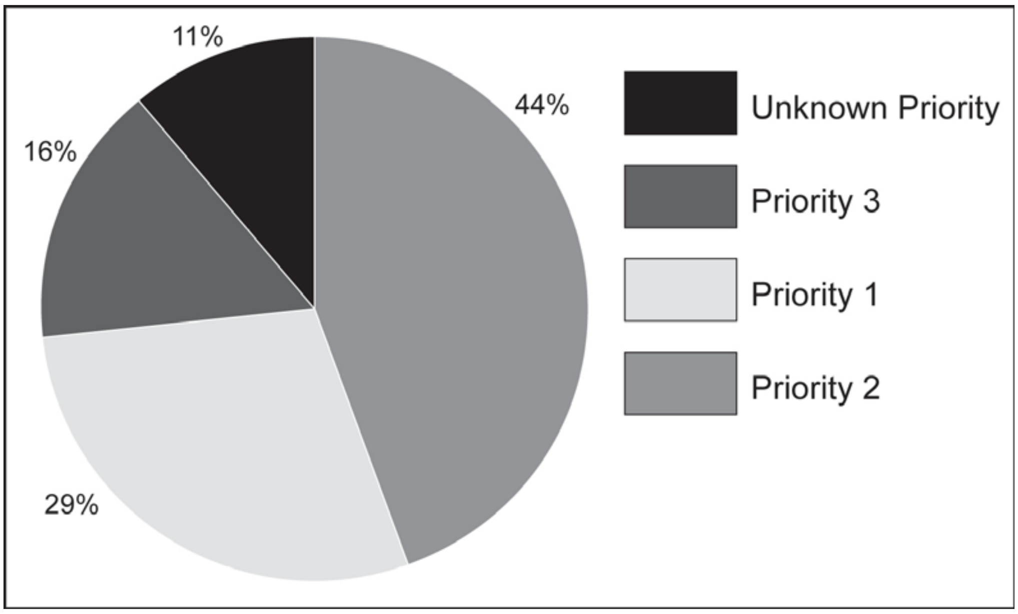


Figure 2.8: Priorities associated with guideline violations. The pie chart shows the percent represented by each category in counter clockwise order from the top left. We used the priorities associated with the W3C and Section 508 guidelines to determine most priorities.

Figure 2.9 shows the number of tools that reported each of the 45 guideline violations. Guidelines were reported by only one tool in most cases (62 percent), which suggests that there is very little consistency in reported errors across tools. Frequently reported errors include: “provide an alternative, HTML-based means of accessing the same functionality” (five tools); “if this is a data table (not used for layout only), identify headers for the table rows and columns” (four tools); and "if style sheets are ignored or unsupported, are pages still readable and usable" (three tools). Only the latter violation seemed to be valid for the web page.

The inconsistency that Ivory [2003] found in reported violations is similar to inconsistencies discovered in usability evaluations. As an example, two comparative usability testing studies (CUE-1 [Molich et al., 1998] and CUE-2 [Molich et al., 1999]) demonstrated less than a 1% overlap in findings among four and nine independent usability testing teams when they evaluated the same two user interfaces. The lack of systematicity or predictability in the findings of automated evaluation tools is surprising, especially because they evaluate seemingly similar aspects.

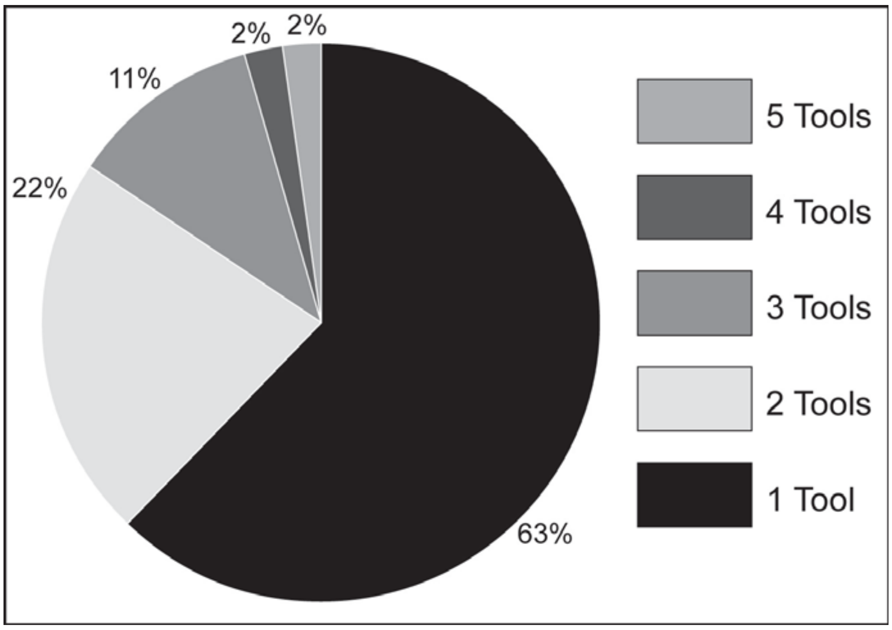


Figure 2.9: Errors reported across tools. The pie chart shows the percent represented by each category in counter clockwise order from the top left.

Errors reported by each tool

Figure 2.10 contrasts the number of violations reported by each of the tools. Some violations were reported as actual errors, while some were reported as alerts (i.e., issues that the practitioner should be aware of, though the tools did not actually detect a problem in the web page). The tools reported an average of eleven violations, seven of which (62 percent) were valid, on average. Bobby and the 508 Accessibility Suite both reported over 100 errors, which is at least five times as many as those reported by the other eight tools. However, only 18–20 percent of the reported errors were unique (i.e., not repeated). Although it could be overwhelming to process such a large number of errors, Bobby simplifies this process by only presenting each error once, along with its number of occurrences. The 508 Accessibility Suite does not aggregate errors in this manner.

Figure 2.9 shows that the remaining tools reported a smaller number of errors, of which, larger percentages were unique, valid, and assessed automatically versus requiring practitioner inspection. AccVerify is an exception to this trend; it only reports two violations, both were invalid and not assessed automatically. Surprisingly, even though the tools support automated assessment, for tools like Bobby and WAVE, only a few violations are checked automatically. Essentially, the tools flag a number of issues, which the practitioner has to investigate to determine if they are valid and address as necessary.

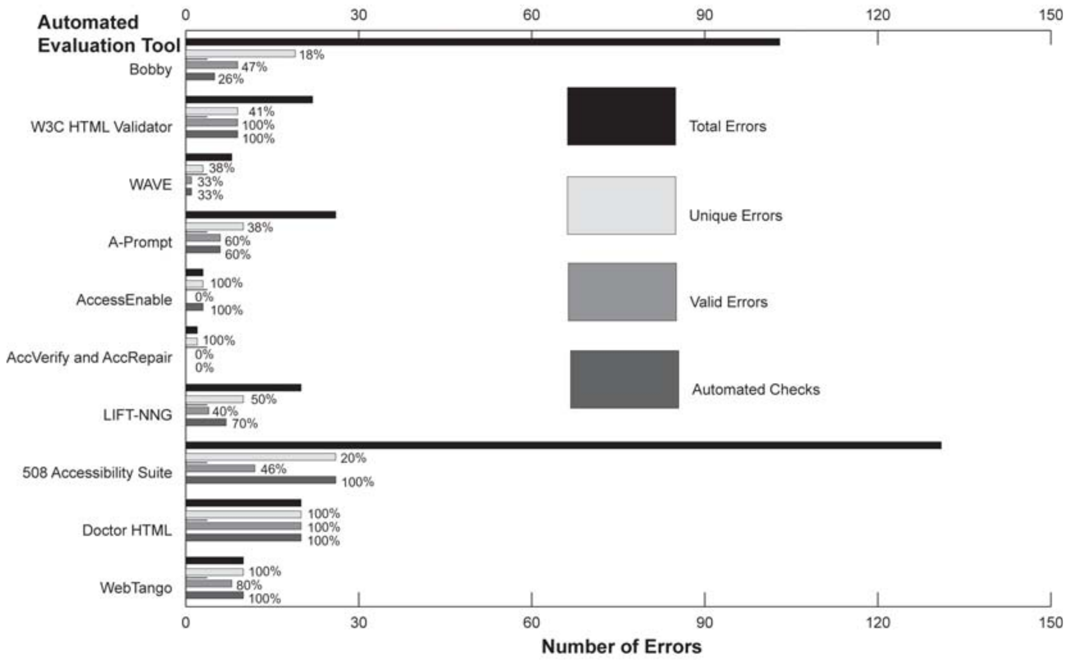


Figure 2.10: Errors reported by each tool. From top to bottom, the total number of violations (errors and alerts), the number of violations that are unique (i.e., not repeated), the number of unique violations that are valid for the web page (Valid Errors), and the number of unique violations that are assessed automatically (Automated Checks) for each tool. Percentages next to the bars for unique errors reflect the ratio of the number of unique errors to the total number of errors. Percentages next to the bars for valid errors reflect the ratio of valid errors to the number of unique errors. Similarly, percentages next to the bars for automated checks reflect the ratio of errors that are assessed automatically to the number of unique errors.

2.5 Expanding existing approaches to automating evaluation by guideline review

Existing guideline review tools appear to address some user abilities in their assessments. However, the emphasis of these tools seems to be heavily on supporting blind users and users with comprehension impairments, at least to some degree. The tools currently provide little or no assessment support for users who are not able to use a keyboard or have hearing impairments. The tools do not appear to be robust enough to handle non-HTML implementations of web sites, such as implementations in Macromedia Flash, dynamic HTML, and other technologies. Furthermore, one of the major limitations of these tools is that they do not analyze content, even though content is considered the most important aspect of a web site [Sinha et al. 2001].

With few exceptions, all the tools were easy to use and their reports were straightforward to read. The comparison of the assessments that the ten guideline review tools generated for the same web page showed that there was little consistency in evaluations across the tools, even though most tools are based on the common W3C and Section 508 guidelines. Tools such as Bobby and WAVE, simply flag errors for which the practitioner needs to determine if they are valid and, if necessary, to address them accordingly. Nonetheless, Bobby, A-Prompt, and the 508 Accessibility Suite cover broader user abilities in their assessments than the other seven tools.

Even if the tools support a range of user abilities, their effectiveness is unclear. An empirical survey of practitioners [Ivory 2003] revealed that they have mixed feelings about the tools. Furthermore, the analysis shows that there is little consistency in evaluation results across tools for the same web page. It also showed that less than half of the reported errors were invalid.

2.5.1 Anatomy of existing evaluation tools

Ivory [2003] attributed these issues to the fact that the tools use separate evaluation engines (even for the same guidelines), rather than employing a common evaluation core.

Given that existing automated evaluation tools are based on similar automated evaluation methodologies, there is a lot of similarity across tools. Figure 2.11 depicts the anatomy of tools. They have internal assumptions about web site users (e.g., computer or Internet experience, reading level, and other abilities), their tasks (e.g., browsing or searching for information), and the technology that they use (e.g., web browsers, Internet connections, and assistive technology). For example, the tools may assume that the users are sighted, browsing for information, and accessing the site via a computer with a 56.6K modem and a 15-inch monitor. As another example, the tools may assume that site users are blind, browsing for information, and accessing the site via a computer with a 56.6K modem and screen reader. It is important to note that these assumptions may not match the user characteristics that influenced the design of the site. All tools have evaluation criteria (e.g., server response time or guidelines) that they use to evaluate a site, and they typically present evaluation results in report format (graphical or textual).

In addition, most existing evaluation tools adopt the same evaluation approach (Figure 2.12):

- The evaluation logic is hard coded inside the evaluation engine as one or more procedures that will be sequentially called during the evaluation process.
- The main user input is the URL of the evaluated page (on-line or locally).
- An evaluation report is generated listing identified usability problems. The clarity and the utility of this report may vary from one tool to another.
- The parsing of the web page is generally independent of the guidelines considered and could cause the page to be parsed more than once to generate a DOM-like structure of the page.

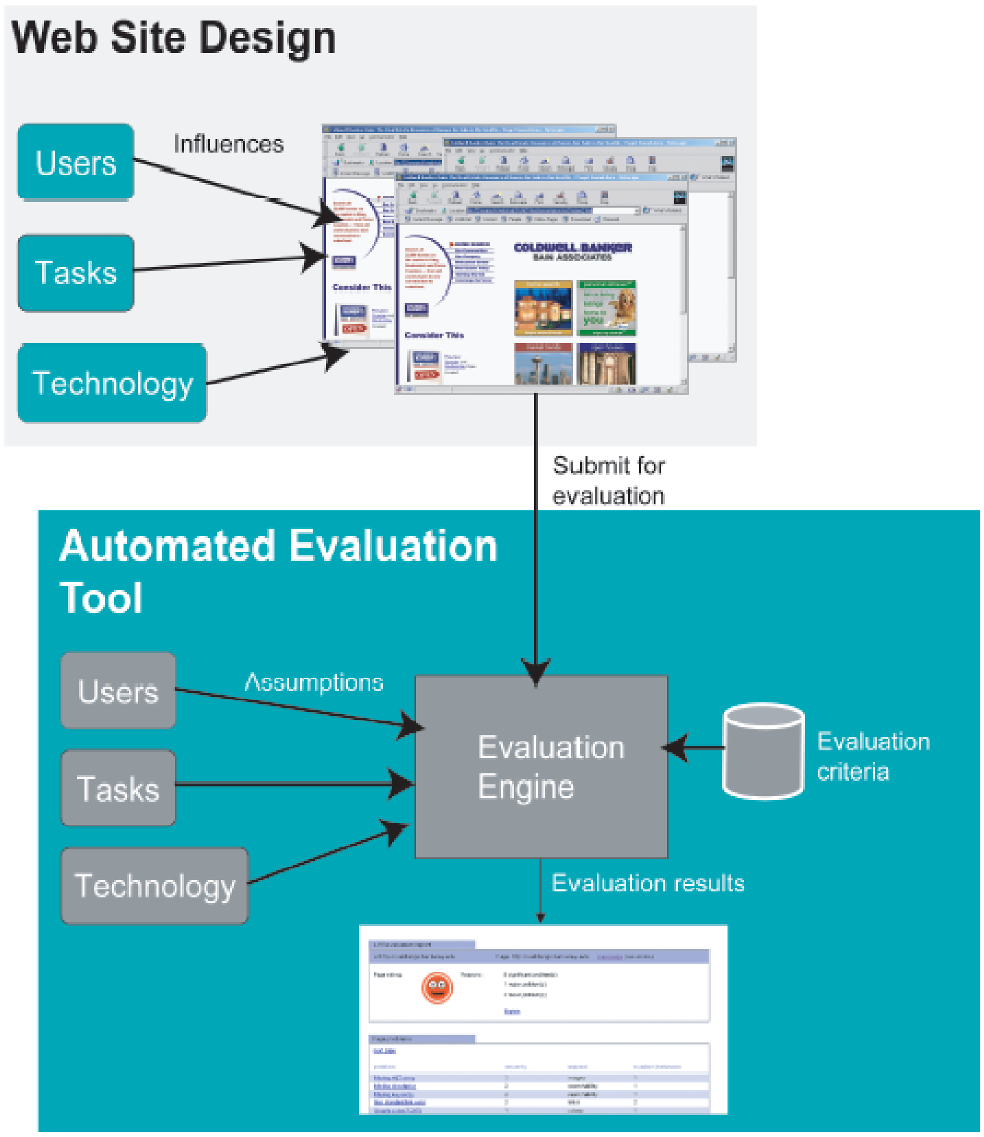


Figure 2.11: Anatomy of an automated evaluation tool. The top part of the figure shows that a web site design is influenced by the intended users, their tasks, and assumptions about the technology that they will use to access the site. The site is submitted to an evaluation tool for assessment. The tool has internal assumptions about the users, tasks, and technology used on the site; these assumptions may or may not match the intended site users. The tool uses these assumptions and its evaluation criteria to evaluate the site and then outputs assessment results, typically in report format.

Advantages of the approach

- The implementation of the evaluation procedure is optimal because the developer is totally concentrated on one guideline at a time.
- The evaluation is generally rapid because everything is already provided in the procedures. The evaluation process is efficient since everything is implemented in straightforward procedures and functions code.
- Minimal user intervention which is suitable for non expert users.

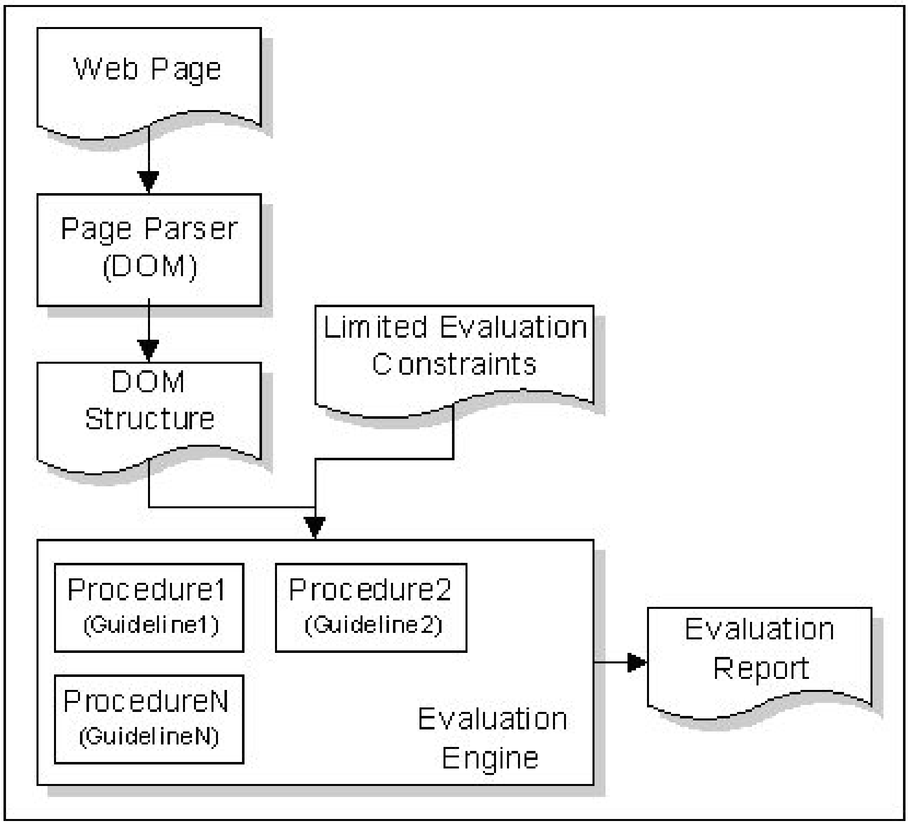


Figure 2.12: Simplistic software architecture of existing Web automatic evaluation tools

Disadvantages of the approach

- As evaluation logic of guidelines is expressed in form of internal procedures, it is impossible to add a new guideline, to modify or to delete an existing one without modifying the source code of the evaluation too. This is the main reason for the tools inflexibility in facing the urging need to follow continuously changing usability guidelines in the internet world.

For example, the WCAG1.0 [W3C 1999] is currently under review and a new set of guidelines is being developed that refine and clarify the original set. This is due to the fact that WCAG1.0 were general enough that they covered all aspects of web content to some degree but there was a level of detail missing that left many guidelines open to wide interpretation. In the development of software tools that evaluated web content for compliance to the guidelines it became apparent that the guidelines needed more detailed specifications. An obvious example is that the guidelines specify that all images used on web pages must have alternate ("alt") text so that users unable to view the images know the image purpose. The specification for valid "alt" text was not described in the WCAG1.0 which left many people wondering if their "alt" text would pass the guidelines. Could empty "alt" text or "alt" text that was all spaces be valid? Another example is the guideline about color use in Web pages. "Sufficient contrast" was a term used in the WCAG1.0 to describe foreground and background color combinations that needed detailed definition, and it was not possible even for (good) UE tools to evaluate it automatically. The ATRC [Ridpath and Treviranus 2002] undertook a study on the web to determine the values that people subjectively describe as good or bad color contrast combinations. Using the results of this study they created an algorithm that can determine if the colors used on a web page give sufficient contrast for most people to read text. Thus, existing UE tools like LIFT will have to modify their code to implement this algorithm.

- The guidelines review process is not very efficient, thus leading to redundant assessments. For instance, several WCAG1.0 and Section508 guidelines share similar checkpoints for different elements.
- Potentially, we need more than one evaluation tool to cover all the desired guidelines.

2.5.2 A new approach for automated Web UE

This section presented an overview of major usability evaluation tools that conduct evaluation by guideline review, and it underlined advantages and shortcomings of this class of tools. Based on tools surveyed and the findings of Ivory's studies, it is **our opinion that these shortcomings are mainly due to hard coding the evaluation logic inside the evaluation engine**. For this reason, we think that structuring guidelines in a consistent and systematic manner outside the evaluation tool would overcome these shortcomings and offer a lot of advantages:

- It should allow the dynamic evaluation of any guideline without any code modification of the evaluation tool.
- It should enable an evaluator to manipulate the supported guidelines: add, delete, and redefine evaluation logic of a guideline if needed, without the need to modify and recompile the evaluation engine.
- It should enable the evaluator to realize on-demand evaluation by providing him by high control level of the evaluation process.
- As we will see in chapter 5, the richness of the evaluation tool depends on the underlying GDL: if the GDL grammar is rich enough, our tool should be able to evaluate any guideline as soon as there are HTML elements that permit its evaluation. Obviously, a minimal requirement of our methodology is to be able to evaluate any guideline evaluable by existing UE tools.
- Using a XML-compliant form to structure guidelines would enable a kind of UE portability: we can imagine an online evaluation service that can evaluate guidelines structured according to our GDL. This service can use guidelines stored in the service central XML database located on the same server or in a local XML database where the evaluator stored his own guidelines after structuring them in a GDL-compliant form. This scenario is practically feasible with few configuration parameters.

In the next chapter we present our methodology for automating Web U&A evaluation by guideline review. This methodology is based on a framework for structuring guidelines in a systematic and consistent manner (Chapter 4), and a formal language for expressing these structures in a XML-compliant form (Chapter 5).