Automated Usability Evaluation of WIMP and Web Interfaces

Melody Y. Ivory
EECS Department
UC Berkeley
Berkeley, CA 94720-1776 USA

Marti A. Hearst
SIMS
UC Berkeley
Berkeley, CA 94720-4600 USA

ABSTRACT

Usability evaluation is an increasingly important part of the iterative design process. Automated usability evaluation (AUE) has great promise as a way to augment existing evaluation techniques, but is greatly underexplored. We present a new taxonomy for automated usability analysis and illustrate it with an extensive survey. We discuss ways to expand the application of AUE to WIMP and Web interfaces.

Keywords

Automated Evaluation, User Interface, Web, Taxonomy

INTRODUCTION

Usability evaluation (UE) is an increasingly important activity to ensure that user interfaces (UIs) enable users to achieve specific goals effectively, efficiently and satisfactorily. The rise of the Web changes the usability assessment landscape, enabling evaluators to reach a much wider audience and allowing for user testing on a larger scale than is economically feasible for standard graphical interfaces. Nonetheless, usability evaluation is still a major bottleneck in the iterative design process, and innovations to support this activity lag far behind support for designing and implementing traditional and Web UIs [14, 32, 38].

How can usability evaluation be improved? One approach is to expand the use of *automated* usability evaluation (AUE) methods. Automated methods should be useful for aiding in comparisons between alternative designs, and for predicting time and error costs across an entire design. It is important to note that we consider automated techniques to be a useful *addition* to standard evaluation techniques such as heuristic evaluation and user testing – not a substitute – because these techniques uncover different kinds of problems [19, 29, 30].

In this paper we present a taxonomy for clearly expressing UE automation and illustrate it with an extensive survey of automated methods. This survey includes automated inquiry, inspection, testing, analytical modeling and simulation methods applied to both WIMP (Windows, Icons, Menus & Pointers) and Web interfaces. The results of this survey indicate promising directions for future work in automated usability analysis.

A TAXONOMY OF AUE

Several surveys of UE methods for WIMP interfaces exist; Hom [13] and Zhang [15] provide a detailed discussion of inspection, inquiry and testing methods. Several taxonomies of UE methods have also been proposed. The most commonly used taxonomy is one that distinguishes between predictive (e.g., GOMS analysis and cognitive walkthrough) and experimental (e.g., user testing) techniques [8]. Whitefield et al. [45] present another classification scheme based on the presence or absence of a user and a computer. Neither of these taxonomies reflect the automation aspects of UE methods.

The sole existing survey of automated usability evaluation, by Balbo [2], uses a taxonomy which distinguishes among four features of automation:

- **None** - no level of automation supported (i.e., evaluator performs method).
- **Automatic Capture** - software automatically captures interface usage (e.g., logging).
- **Automatic Analysis** - automatic identification of usability problems.
- **Automatic Critic** - automatic analysis coupled with automated suggestions for improvements.

Balbo uses these categories to classify 10 common and uncommon UE methods. However most of the methods surveyed require extensive human effort (because they rely on formal user testing and/or require extensive evaluator interaction). For example, Balbo classifies using log files as an automatic analysis method, but this method requires formal user testing sessions to generate those log files. What Balbo calls an automatic critic method may require the evaluator to create a complex model as input. Thus, this classification scheme is somewhat misleading since it ignores the non-automated requirements of the UE methods.

We expand this taxonomy to include consideration of a method's non-automated testing requirements, both in terms of users and evaluators. We augment each of

Balbo's features with an attribute called **testing level**; this indicates the human testing effort required for execution of the method:

- **None** - does not require testing or modeling.
- **Formal** - requires a user or evaluator to complete a set of structured tasks and/or a procedure.
- **Informal** - requires normal use (i.e., unstructured tasks completed by a user or evaluator).
- **Model** - requires the evaluator to develop a UI model and/or a user model in order to employ the UE method.

These testing levels are discussed in more detail below. To organize the discussion, we group existing UE methods into 5 general classes: **testing, inquiry, inspection, analytical modeling** and **simulation**. Software engineering practices have had a major influence on the first three classes, while the latter two, analytical modeling and simulation, are quite similar to performance evaluation techniques used to analyze the performance of computer systems [18, 16].

In summary, our taxonomy consists of: a UE method type (testing, inquiry, inspection, analytical modeling and simulation); an automation type (none, capture, analysis and critic); and a testing level (none, formal, informal and model). In the remainder of this paper, we use this taxonomy to analyze UE evaluation methods.

ANALYSIS OF UE METHODS

We surveyed 56 UE methods applied to WIMP interfaces, and 50 methods applied to Web interfaces. The automation patterns are similar for both, with the exception that analytical modeling and simulation methods are far less explored in the Web domain than for WIMP interfaces (5 vs. 15 methods).

There are major differences in automation among the 5 types of methods. Table 1 depicts automation characteristics for surveyed methods for WIMP and Web interfaces [2, 13, 15, 31, 35, 47]. We note each method's automation type and its testing level. For some methods, we will discuss more than one approach; hence, we add the number of methods surveyed in parenthesis beside the testing level.

Table 1 shows that AUE in general is greatly underexplored. If we sum the methods by automation type, we see that non automatic methods represent 65% of the methods surveyed, while automated methods collectively represent only 35%. Of this 35%, automatic capture methods represent 15%, automatic analysis methods represent 18% and automatic critic methods represent 2%. All of the automatic capture and log file analysis methods require some level of testing. Hence, only 18% of the automated methods do not require formal or informal testing to employ.

To be fully automated, an AUE method would provide the highest level of automation (i.e., critic) and require no testing. Our survey found that this level of automation has been accomplished using only one method: guideline reviews [26]. Operationalized guidelines automatically detect and report usability violations and then make suggestions for fixing them.

Of those methods that support the next level of automation (i.e., analysis), Table 1 shows that analytical modeling and simulation methods represent the majority. These methods support automatic analysis without requiring formal or informal user testing. All of these methods embed analysis within the design phase of UI development, as opposed to after development. This is an important point, since most evaluation is done after the interface has been built [31]. The next sections discuss the various UE types, and their existing levels of automation, in more detail. Methods that support only capture without analysis or critique (found under user testing and inquiry methods in Table 1) have only a minor amount of automated aspects and so are not discussed in detail here. A more complete survey can be found in [17].

User Testing Methods

Usability testing with real participants is one of the most fundamental usability evaluation methods [31]. It provides an evaluator with direct information about how people use computers and what their exact problems are with the interface being tested. During usability testing, participants use the system or a prototype to complete a pre-determined set of tasks while the tester records the results of the participants' work. The tester then uses these results to determine how well the interface supports users' completing their tasks.

Automation has been used predominantly in two ways within the user testing class: automatically capturing use data and automatically analyzing it according to some metrics or a model (referred to as log file analysis in Table 1). In rare cases methods support both automatically capturing and analyzing use data [27].

Log file analysis methods support automatic analysis of data logged during user testing. The Web enables user testing on a much larger scale than is economically feasible with WIMP interfaces. Hence, log file analysis is a heavily used methodology for evaluating Web interfaces. Our survey revealed four general approaches for analyzing WIMP and Web log files: metric-based, pattern-matching, task-based, and inferential.

Metric-based Analysis of Log Files. Metric-based approaches generate quantitative performance measurements. Two such examples for WIMP interfaces are DRUM and the MIKE UIMS (User Interface Management System) [33]. DRUM outputs measurements that reflect effectiveness, efficiency and learnability and de-

UE Method	Automation Type				Description
	None	Capture	Analysis	Critic	
Testing (Formative)					
Thinking-aloud Protocol	(1)				user talks during test
Question-asking Protocol	(1)				tester asks user questions
Shadowing Method	(1)				expert explains user actions
Coaching Method	(1)				user can ask an expert questions
Teaching Method	(1)				user teaches novice
Co-discovery Learning	(1)				two users collaborate
Performance Measurement	(1)	F (5)			capture quantitative data
Log File Analysis			FIM (13)*		analyze captured usage data
Retrospective Testing	(1)				review videotape with user
Remote Testing		F (1)			distance testing
Inspection (Formative)					
Guideline Reviews	(5)		(2)	M (2)	guideline conformance
Cognitive Walkthrough	(1)	F (1)			simulate problem solving
Pluralistic Walkthrough	(1)				group cog. walkthrough
Heuristic Evaluation	(1)				identify heuristic violations
Perspective-based Inspection	(1)				narrowly focused heur. eval.
Feature Inspection	(1)				evaluate product features
Formal Usability Inspection	(1)				formal heur. eval.
Consistency Inspection	(1)				UI consist. across products
Standards Inspection	(1)				industry standard compliance
Inquiry (Summative)					
Contextual Inquiry	(1)				field interviewing
Field Observation	(1)				observe system use
Focus Groups	(1)				user group discussion
Interviews	(1)				formally ask user questions
Surveys	(1)	I (1)			informal interview
Questionnaires	(1)	I (1)			subjective evaluation
Journaled Sessions		FI (1)			UI logs system use
Self-reporting Logs	(1)				user records UI operations
Screen Snapshots	(1)				user captures UI screens
User Feedback	(1)				user-initiated comments
Analytical Modeling (Predictive)					
GOMS Analysis			M (4)		execution & learning time
UIDE Analysis			M (3)		analysis within a UIDE
HTML Authoring Analysis			(2)		analysis within authoring tool
Programmable User Models			M (1)		programming UI to fit user
Simulation (Predictive)					
Information Processor Model			M (9)		simulating user interaction
Automation Type					
Total	26	6	7	1	
Percent	65%	15%	18%	2%	
Total Methods Surveyed	30	10	34	2	

Table 1: Automation characteristics of WIMP and Web UE methods. A number in parentheses indicates the number of methods surveyed for a particular method and automation type. The testing level for each method is represented as: none (blank), formal (F), informal (I) and model (M). The * for the FIM entry indicates that either formal or informal testing is required. In addition, a model may be used in the analysis.

tects critical incidents specified by the evaluator. The MIKE UIMS generates a number of general, physical, logical and visual metrics, including performance time, command frequency, the number of physical operations required to complete a task, and required changes in the user's focus of attention on the screen.

For the Web, Service Metrics [28] offers SM-WEBPOINT that allows evaluators to pinpoint performance bottlenecks, such as slow server response time, that may negatively impact the usability of a Web site. Service Metrics also offers SM-WEB that employs a network of measurement agents to collect similar metrics as SM-WEBPOINT from multiple geographical locations under various access conditions.

Pattern-Matching Analysis of Log Files. Pattern-matching approaches, such as MRP (Maximum Repeating Pattern) [39] and ÉMA (Automatic Analysis Mechanism for the Ergonomic Evaluation of User Interfaces) [3], analyze user behavior captured in logs. MRP detects and reports repeated user actions (e.g. consecutive invocations of the same command) that may indicate usability problems. ÉMA, which is also a task-based approach, extends the MRP approach to detect additional behavior patterns, such as immediate task cancellation.

USINE (USer Interface Evaluator) [24], which is also a task-based approach, employs the ConcurTaskTrees [34] notation to express temporal relationships among UI tasks (e.g., enabling and disabling). Using this additional information, USINE looks for precondition errors (i.e., task sequences that violate temporal relationships) and also reports quantitative metrics (e.g., task completion time) and information about task patterns, missing tasks and user preferences.

Task-based Analysis of Log Files. Task-based approaches, such as ÉMA and USINE, analyze discrepancies between the designer's task model and actual use. ÉMA uses a data-flow task model along with behavior heuristics to detect specific user behaviors that may indicate usability problems. Immediate task cancellation or a shift in direction during task completion are two behaviors detected by ÉMA.

The Quantitative User Interface Profiling (QUIP) [12] tool provides one of the most advanced approaches to task-based, log file analysis and visualization for Web UIs. Unlike other approaches, QUIP aggregates traces of multiple user interactions and compares the task flows of these users to the designer's task flow. QUIP encodes quantitative time- and trace-based information, such as the average time between actions and the fraction of users who performed a particular sequence of actions into directed graphs.

Inferential Analysis of Log Files. Inferential analysis of Web log files includes traffic-based analysis (e.g., pages-per-visitor or visitors-per-page) and time-based analysis (e.g., click paths and page-view durations) [10, 11, 41]. These techniques attempt to infer something about usability or the user's interest in page contents from log files. This analysis is largely inconclusive, since server logs provide only a partial trace of user behavior. Furthermore, server log files are missing valuable information about what tasks users want to accomplish [5].

Inspection Methods

During a usability inspection, the evaluator inspects or examines usability aspects of a UI using an evaluative criteria, and then gives feedback on potential usability problems. Unlike other UE methods, inspections rely solely on the evaluator's judgement as a source of evaluation feedback, and thus are considered to be informal methods.

Automation has been predominately used to check guideline conformance. KRI/AG (Knowledge-based Review of user Interface) [26] is one such tool that automatically checks the guideline conformance of X Window UI designs created with the TeleUSE UIMS. KRI/AG contains a knowledge base of guidelines and style guides. It uses this information to automatically critique a UI design and generate comments about possible flaws in the design. SYNOP [2] is a similar automatic critic system that performs a rule-based critique of a control system application. SYNOP automatically modifies the UI model based on its evaluation.

Guideline reviews are also actively used to evaluate Web interfaces. Many corporations develop their own guidelines and there have been several efforts to develop a standard set of guidelines. Keevil [21] presents a set of guidelines as a list of yes/no questions about the site organization, user-oriented tasks and technical content of a Web site. After answering these questions in a spreadsheet or Web form, a usability index can be computed for a site. Ambuhler and Lindenmeyer [1] use guidelines to compute accessibility measurements (i.e., how easy it is to access a page without special hardware or software) for a Web site. Lohse and Spiller [25] use regression modeling on a set of 32 guidelines to predict store traffic and dollar sales as a function of interface features, such as the number of links into the store and number of products. Finally, Rossi et al. [36] propose guidelines to assist designers with determining the best navigation structure for a site. Currently, all of these approaches require manual implementation and evaluation.

There are two automatic analysis tools that use guidelines for usability checks. The first is the Web Static Analyzer Tool (SAT) [37], part of the NIST WebMetrics suite of tools. It assesses static HTML according to a number of guidelines, such as whether all graphics contain ALT tags and the average number of words in link text. Future plans for this tool include adding the ability to look at the entire site at once to identify potential usability problems in interactions between pages. A similar automated analysis tool is The Rating Game [40]. It is also a static HTML analysis approach based on guidelines.

Analytical Modeling Methods

Analytical modeling complements other evaluation techniques like user testing. Given some representation or model of the UI and/or user, these methods inexpensively generate quantitative usability predictions. Automation has been predominately used to analyze task completion (e.g. execution and learning time) within WIMP UIs and Web site structure (e.g. breadth and depth).

Most analytical modeling approaches for WIMP UIs are based on the model human processor (MHP) proposed by Card et al. [7]. GOMS analysis is one of the most widely accepted analytical modeling methods based on the MHP [20]. Other methods based on the MHP employ simulation and will be discussed in the next section.

The GOMS family of analytical methods use a task structure consisting of Goals, Operators, Methods and Selection rules. Using this task structure along with validated time parameters for each operator, the methods predict task execution and learning times for error-free expert performance. The four approaches in this family include the original GOMS method proposed by Card, Moran and Newell (CMN-GOMS) [7], the simpler keystroke-level model (KLM), the natural GOMS language (NGOMSL) and the critical path method (CPM-GOMS) [20]. These approaches differ in the task granularity modeled (e.g., keystrokes or a high-level procedure) and in the support for alternative methods (i.e., selections) and multiple goals.

Two of the major roadblocks to using GOMS have been the tedious task analysis and calculation of execution and learning times that are usually manually performed. USAGE (the UIDE System for semi-Automated GOMS Evaluation) [6] and CRITIQUE (the Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations) [14] are tools that address these limitations by automatically generating a task model and quantitative predictions for the model. Both of these tools accomplish this within a user interface development environment (UIDE). GLEAN (GOMS Language Evaluation and ANalysis) [23] is another tool that generates quantitative predictions for a given GOMS task model (discussed in more detail in the next section).

While USAGE and CRITIQUE focus on GOMS analysis within a UIDE, there is a class of methods that support metric-based analysis of UI designs within a UIDE. AIDE (semi-Automated Interface Designer and Evaluator) [38] is one example that helps designers assess and compare different design options using quantitative task-sensitive and task-independent metrics, including efficiency, alignment, horizontal balance and vertical balance. AIDE also generates initial UI layouts.

Programmable user models (PUM) [46] is an entirely different analytical modeling technique for automatic analysis. For this approach, a UI designer programs a psychologically constrained architecture to simulate a user performing a range of tasks in the proposed UI. The architecture places limitations on the amount of information that must be recalled by a user and requires the designer to specify explicit sequences of operations for each task. Difficulties experienced by the designer while programming the architecture can then be used to improve the UI. Once the designer successfully programs the architecture, the program can be executed to generate quantitative results.

Analytical modeling of Web UIs lags far behind efforts for WIMP interfaces. Many Web authoring tools, such as Microsoft FrontPage and Macromedia Dreamweaver, provide limited support for usability evaluation in the design phase (e.g., predict download time and check HTML syntax). HyperAT [42] and Gentler [44] are two research systems from Middlesex University that incorporate more advanced usability evaluation techniques into authoring tools.

The Hypertext Authoring Tool (HyperAT) [42, 43] is a Macintosh prototype for authoring well-structured hyperdocuments and reverse-engineering existing Web sites. HyperAT supports two methods of usability evaluation - structural and log file analysis. The structural analysis verifies that the breadths and depths at both the page and site level fall within thresholds. The log file analysis supports automatic parsing and inferential analysis of server log files. The authors also propose incorporating simulations of user behavior, but have not yet pursued this option. Gentler [44] provides similar structural analysis but focuses on maintenance of existing sites rather than design of new ones.

Simulation Methods

Similar to analytical modeling, simulation complements traditional UE methods and inherently supports automatic analysis. Using models of the user and/or UI, these approaches simulate the user interaction and report the results of this interaction. Evaluators can run simulations with different parameters in order to study various UI design tradeoffs and thus make more informed decisions about UI implementation.

Automation has been used predominately to simulate task completion within WIMP UIs and navigation within Web sites. All of the WIMP simulations rely on some variation of a human information processor model similar to the MHP previously discussed. Pew and Mavor [35] provide a detailed discussion of this type of modeling and an overview of many of these approaches, including five that we discuss: ACT-R (Adaptive Control of Thought), COGNET (COGnition as a NEtwork of Tasks), EPIC (Executive-Process Interactive Control), HOS (Human Operator Simulator) and Soar. Here we also consider CCT (Cognitive Complexity Theory) [22], ICS (Interacting Cognitive Subsystems) [4] and GLEAN (GOMS Language Evaluation and ANalysis) [23]. Rather than describe each method individually, we summarize the major characteristics of these simulation methods below.

Modeled Tasks. The models we surveyed simulate the following 3 types of tasks: a user performing cognitive tasks (e.g., problem-solving and learning: COGNET, ACT-R, Soar, ICS); a user immersed in a human-machine system (e.g., an aircraft and tank: HOS); and a user interacting with a typical UI (EPIC, GLEAN, CCT).

Modeled Components. Some simulations focus solely on cognitive processing (ACT-R, COGNET) while others incorporate perceptual and motor processing as well (EPIC, ICS, HOS, Soar, GLEAN, CCT).

Component Processing. Task execution is modeled either as serial processing (ACT-R, GLEAN, CCT), parallel processing (EPIC, ICS, Soar), or semi-parallel processing (serial processing with rapid attention switching among the modeled components, giving the appearance of parallel processing: COGNET, HOS).

Model Representation. To represent the underlying user, simulation methods use either task hierarchies (as in GOMS task structure: HOS, CCT), production rules (CCT, ACT-R, EPIC, Soar, ICS), or declarative/procedural programs (GLEAN, COGNET). CCT uses both a task hierarchy and production rules to represent the user and system models respectively.

Predictions. The surveyed methods return a number of simulation results, including predictions of task performance (EPIC, CCT, COGNET, GLEAN, HOS, Soar), memory load (ICS, CCT), learning (ACT-R, SOAR, ICS, GLEAN, CCT), or behavior predictions such as action traces (ACT-R, COGNET, EPIC).

WebCriteria's Site Profile [9] is the most advanced simulation method applied to Web interfaces. Its analysis is performed in four phases: gather, model, analyze and report. During the gather phase, a spider traverses a site (200-600 unique pages) to collect Web site data. This data is then used to construct a model of the site based on graph theory. For the analysis phase, it uses a standard Web user model (called Max) to simulate a user's information seeking behavior; this model is based on prior research with GOMS analysis. Given a starting point in the site, a path and a target, Max "follows" the path from the starting point to the target and logs measurement data. These measurements are used to compute an accessibility metric which is then used to generate a report. This approach can be used to compare Web sites, provided that an appropriate navigation path is supplied for each.

DISCUSSION

Even if the time is taken to do a usability assessment, it can be difficult to perform successfully (in part due to the wide variation in the dependent variables, including users, tasks, and interface designs). Several studies have illustrated this difficulty by showing that UE findings can vary widely when different evaluators study the same UI using different and similar methods [19, 29, 30, 31]. In particular, two comparative studies (CUE-1 [29] and CUE-2 [30]) were performed on a Windows calendar management application and the Hotmail Web site. These studies made use of 4 and 8 independent usability evaluation teams, respectively. In both studies, there was less than a 1% overlap in findings among the teams.

Although all of the usability problems found are valid and useful, this result implies a lack of systematicity or predictability in the findings of usability evaluations. This could be addressed by employing multiple usability teams during evaluation similar to the comparative studies. However, usability assessment with only one team is expensive. AUE methods could be inexpensively employed to provide complementary usability data and to widen UI coverage. We discuss several promising AUE techniques below.

Our survey showed log file analysis to be a viable methodology for automated analysis of usage data. However, it still requires formal or informal testing to employ. One way to expand the use and benefits of this methodology is to leverage a small amount of test data to generate a larger set of plausible usage data. We discuss this option in more detail in [16]. This is even more important for Web interfaces, since server logs do not capture a complete record of user interactions.

Given a wider sampling of usage data, task-based log file analysis is a promising research area to pursue. Task-based approaches that follow the USINE model in particular (i.e., compare a task model expressed in terms of temporal relationships to usage traces) provide the most support, among the methods surveyed, for understanding user behavior, preferences and errors. Although the authors claim that this approach works well for WIMP UIs, it needs to be adapted to work for Web UIs where tasks may not be clearly-defined. Additionally, since USINE already reports substantial analysis data, this data could be compared to usability guidelines in order to support automated critique.

Our survey also showed that analytical modeling within a UIDE is a promising approach for automated analysis. The AIDE approach provides the most support for evaluating and improving UI designs and could be expanded to Web interfaces. Guidelines could also be incorporated into AIDE analysis to support automatic critic. Although UIDE analysis is promising, it is not widely-used in practice. Applying such approaches outside of these environments is an open research problem. We discuss other promising analytical modeling approaches based on performance evaluation in [16].

Finally, our survey showed that existing simulations based on a human information processor model have widely different uses. Thus, it is difficult to draw concrete conclusions about the effectiveness of these approaches. Simulation in general is a promising research area to pursue for AUE, especially for evaluating alternative designs. We discuss promising simulation approaches based on performance evaluation in [16].

CONCLUSIONS/FUTURE WORK

In this paper we presented a taxonomy for clearly expressing UE automation. This taxonomy reflects the testing requirements of each method. We also presented an extensive survey of AUE methods for WIMP and Web interfaces. We found that AUE methods represent only 35% of methods surveyed. Of these methods only 18% (WIMP) do not require formal or informal testing to employ. Of these, all but one are based on analytical modeling or simulation. We discussed how research to further develop analytical modeling, simulation and log file analysis techniques could enhance traditional UE methods.

This survey suggests a viable direction to follow towards developing an automatic UE method for evaluating information-centric Web sites. Based on these findings, we intend to pursue developing a simulation methodology and tool to help designers explore design alternatives and improve information architectures prior to web site implementation. We discuss a framework for this methodology in [16] to complement traditional evaluation methods.

REFERENCES

1. Reto Ambühler and Jakob Lindenmeyer. Measuring accessibility. In *Proceedings of WWW8*, 1999.
2. Sandrine Balbo. Automatic evaluation of user interface usability: Dream or reality. In *Proceedings of QCHI 95*, 1995.
3. Sandrine Balbo. ÉMA: Automatic analysis mechanism for the ergonomic evaluation of user interfaces. Technical Report 96/44, DSIRO Division of Informaiton Technology, 1996.
4. Philip J. Barnard. Cognitive resources and the learning of human-computer dialogs. In John M. Carroll, editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, pages 112–158. The MIT Press, 1987.
5. Michael D. Byrne, Bonnie E. John, Neil S. Wehrle, and David C. Crow. The tangled web we wove: A taxonomy of WWW use. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*, volume 1, pages 544–551, 1999.
6. Michael D. Byrne, Scott D. Wood, Piyawadee "Noi" Sukaviriya, James D. Foley, and David Kieras. Automating interface evaluation. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1, pages 232–237, 1994.
7. Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
8. J. Coutaz. Evaluation techniques: Exploring the intersection of HCI and software engineering. In *Proceedings of the International Conference on Software Engineering*, 1994.
9. Web Criteria. Max, and the objective measurement of web sites. http://www.webcriteria.com, 1999.
10. M. Carl Drott. Using web server logs to improve site design. In *ACM 16th International Conference on Systems Documentation*, pages 43–50, 1998.
11. Rodney Fuller and Johannes J. de Graaff. Measuring user motivation from server log files. In *Proceedings of the Human Factors and the Web 2 Conference*, October 1996.
12. Brian Helfrich and James A. Landay. QUIP: qunatitative user interface profiling. Unpublished manuscript, 1999.
13. James Hom. The usability methods toolbox. http://www.best.com/∼jthom/usability/usable.htm.
14. Scott E. Hudson, Bonnie E. John, Keith Knudsen, and Mickael D. Byrne. A tool for creating predictive performance models from user interface demonstrations. In *Proceedings of the ACM Symposium on User Interface Software and Technology, To appear*, 1999.
15. U S WEST Communications Human Factors Engineering. Usability evaluation methods. http://www.cs.umd.edu/∼zzj/UsabilityHome.html.
16. Melody Y. Ivory and Marti A. Hearst. Comparing performance and usability evaluation: New methods for automated usability assessment. Submitted for publication, 1999.
17. Melody Y. Ivory and Marti A. Hearst. A comprehensive survey of performance evaluation and usability evaluation methodologies. Unpublished manuscript, 1999.
18. R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,*. Wiley-Interscience, New York, NY, USA, May 1991.

19. Robin Jeffries, James R. Miller, Cathleen Wharton, and Kathy M. Uyeda. User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 119–124, 1991.

20. Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, 1996.

21. Benjamin Keevil. Measuring the usability index of your web site. In *ACM 16th International Conference on Systems Documentation*, pages 271–277, 1998.

22. David Kieras and Peter G. Polson. An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 22(4):365–394, 1985.

23. David E. Kieras, Scott D. Wood, Kasem Abotel, and Anthony Hornof. GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 91–100, 1995.

24. Andreas Lecerof and Fabio Paternó. Automatic support for usability evaluation. *IEEE Transactions on Software Engineering*, 24(10):863–888, October 1998.

25. Gerald L. Lohse and Peter Spiller. Quantifying the effect of user interface design features on cyberstore traffic and sales. In *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems*, volume 1, pages 211–218, 1998.

26. Jonas Lowgren and Tommy Nordqvist. Knowledge-based evaluation as design support for graphical user interfaces. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 181–188, 1992.

27. Miles Macleod and Ralph Rengger. The development of DRUM: A software tool for video-assisted usability evaluation. In *Proceedings of the HCI'93 Conference on People and Computers VIII*, pages 293–309, 1993.

28. Service Metrics. Service metrics solutions. http://www.servicemetrics.com/solutions/solutionsmain.asp.

29. R. Molich, N. Bevan, S. Butler, I. Curson, E. Kindlund, J. Kirakowski, and D. Miller. Comparative evaluation of usability tests. In *Proceedings of UPA98*, pages 189–200, June 1998.

30. R. Molich, A. D. Thomsen, B. Karyukina, L. Schmidt, M. Ede, W. van Oel, and M. Arcuri. Comparative evaluation of usability tests. In *Proceedings of ACM99*, pages 83–86, May 1999.

31. Jakob Nielsen. *Usability Engineering*. Academic Press, Boston, MA, 1993.

32. Jakob Nielsen and Robert L. Mack, editors. *Usability Inspection Methods*. Wiley, New York, 1994.

33. Dan R. Olsen, Jr. and Bradley W. Halversen. Interface usage measurements in a user interface management system. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, pages 102–108, 1988.

34. F. Paternó, C. Mancini, and S. Meniconi. ConcurTaskTrees: Diagrammatic notation for specifying task models. In *Proceedings of INTERACT '97*, pages 362–369. Sydney: Chapman and Hall, 1997.

35. R. W. Pew and A. S. Mavor, editors. *Modeling Human and Organizational Behavior: Application to Military Simulations*. National Academy Press, Washington, 1998.

36. Gustavo Rossi, Daniel Schwabe, and Fernando Lyardet. Improving web information systems with navigation patterns. In *Proceedings of WWW8*, 1999.

37. Jean Scholtz and Sharon Laskowski. Developing usability tools and techniques for designing and testing web sites. In *Proceedings of the 4th Conference on Human Factors & the Web*, 1998.

38. Andrew Sears. AIDE: A step toward metric-based interface development tools. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 101–110, 1995.

39. Antonio C. Siochi and Deborah Hix. A study of computer-supported user interface evaluation using maximal repeating pattern analysis. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 301–305, 1991.

40. Lincoln D. Stein. The rating game. http://stein.cshl.org/ lstein/rater/, 1997.

41. Terry Sullivan. Reading reader reaction: A proposal for inferential analysis of web server log files. In *Proceedings of the Human Factors and the Web 3 Conference*, June 1997.

42. Yin Leng Theng and Gil Marsden. Authoring tools: Towards continuous usability testing of web documents. In *Proceedings of the 1st International Workshop on Hypermedia Development*, 1998.

43. Yin Leng Theng and Harold Thimbleby. Hyperat: Addressing usability issues in web authoring. In *Proceedings of WebNet '98*, 1998.

44. Harold Thimbleby. Gentler: A tool for systematic web authoring. *International Journal of Human-Computer Studies*, 47(1):139–168, 1997.

45. Andy Whitefield, Frank Wilson, and John Dowell. A framework for human factors evaluation. *Behaviour and Information Technology*, 10(1):65–79, 1991.

46. Richard M. Young, T. R. G. Green, and Tony Simon. Programmable user models for predictive evaluation of interface designs. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, pages 15–19, 1989.

47. Zhijun Zhang, Victor Basili, and Ben Shneiderman. Perspective-based usability inspection: An empirical validation of efficacy. *International Journal of Empirical Software Engineering, special issue on Human-Computer Interaction*, To appear, 1999.