An Empirical Foundation for Automated Web Interface Evaluation

by

Melody Yvette Ivory

B.S. (Purdue University) 1993
M.S. (University of California at Berkeley) 1996

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Marti Hearst, Chair
Professor James Landay
Professor Ray Larson

Fall 2001

An Empirical Foundation for Automated Web Interface Evaluation

Abstract

An Empirical Foundation for Automated
Web Interface Evaluation

by

Melody Yvette Ivory
Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Marti Hearst, Chair

This dissertation explores the development of an automated Web evaluation methodology and tools. It presents an extensive survey of usability evaluation methods for Web and graphical interfaces and shows that automated evaluation is greatly underexplored, especially in the Web domain.

This dissertation presents a new methodology for HCI: a synthesis of usability and performance evaluation techniques, which together build an empirical foundation for automated interface evaluation. The general approach involves: 1. identifying an exhaustive set of quantitative interface measures; 2. computing measures for a large sample of rated interfaces; 3. deriving statistical models from the measures and ratings; 4. using the models to predict ratings for new interfaces; and 5. validating model predictions.

This dissertation presents a specific instantiation for evaluating information-centric Web sites. The methodology entails computing 157 highly-accurate, 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 Web interfaces.

This dissertation presents analysis of quantitative measures for over 5300 Web pages and 330 sites. It describes several statistical models for distinguishing good, average, and poor pages with 93%–96% accuracy and for distinguishing sites with 68%–88% accuracy.

This dissertation describes two studies conducted to provide insight about what the statistical models assess and whether they help to improve Web design. The first study attempts to link expert ratings to usability ratings, but the results do not enable strong conclusions to be drawn. The second study uses the results of applying the statistical models for assessing and refining example sites and shows that pages and sites modified based on the models are preferred by participants – professional and non Web designers – over the original ones. Finally, this dissertation demonstrates use of the statistical models for assessing existing Web design guidelines.

This dissertation represents an important first step towards enabling non-professional designers to iteratively improve the quality of their designs.

To Alluminum and Professo.

Contents

Acknowledgements

First and foremost, all thanks to the most high for giving me the strength, determination, and courage to weather the many storms encountered along the way to completing this dissertation.

I want to thank Alluminum (Allum Ross) and Professo (Angelo Ivory) for putting up with me even at my worst. Your patience, love, and support day in and day out brought me through. Whatever I needed you to do and be, you were there for me. Thank you for taking better care of me than I could possibly take of myself. This dissertation would not be possible without you, so I dedicate it to you. I love you from the bottom of my heart!

I want to acknowledge my advisor, Marti Hearst, for allowing me the freedom to walk this path in my own way while simultaneously shaping and molding me to tap into my full potential. You have been a friend, a mentor, a mother at times, and so much more. You are a tough cookie, but I am glad that I had the privilege of working with you. I want you to know that I truly appreciate you (even your unreasonable demands and high expectations)!

I also want to thank my first advisor, Jim Demmel, for getting me started on this path. I appreciate you taking the risk to have me as your student and teaching me about research. What I learned from our work together played a major role in this dissertation.

I want to acknowledge my other dissertation committee members, James Landay and Ray Larson, for reading this tome in record time so I could bring this journey to an end. I greatly appreciate your support!

I want to thank Professor Kahan for taking interest in my success as a graduate student as well as my effectiveness as a parent. I greatly appreciate your input and guidance over the years.

I want to acknowledge Dr. Sheila Humphreys for being my miracle worker over the years. No matter what crisis came up, you waved your magic wand to make it go away. Even at the last minute when I spilled juice on my laptop and rendered it useless, you loaned me your personal laptop so I could continue pushing forward. I cannot even begin to express my gratitude for your being the wonderful person that you are. How can I ever repay you?

I want to thank Dr. Rashmi Sinha for helping me to lay the foundation for this dissertation. You taught me a lot of what I know about advanced statistics, and this dissertation would not have been possible without your input. Thank you for allowing me to call you with hard questions whenever I needed to. Furthermore, thank you for being a friend even when I was not.

I want to acknowledge all of the GUIR people for your invaluable input over the years. I want to acknowledge the undergraduate and graduate researchers that have contributed to my research over the years: Deep Debroy, Stephen Demby, David Lai, Wai-ling Ho-Ching, Toni Wadjiji, and several others. I want to acknowledge Lincoln Stein for his assistance with an early version of the Metrics Computation Tool, Tom Phelps for his assistance with the extended Metrics Computation Tool, and Maya Draisin and Tiffany Shlain at the International Academy of Digital Arts and Sciences for making the Webby Awards 2000 data available. I want to acknowledge Jared Spool and User Interface Engineering for providing a complimentary copy of the *Designing Information-Rich Web Sites* report series and Doug van Duyne and others at NetRaker for making it possible to use their Web research tools. Finally, I want to acknowledge other researchers in the HCI community who have provided valuable input into this research, including Ben Shneiderman, Jean Scholtz, Sharon Laskowski, B.J. Fogg, and Jean Vanderdonckt.

I want to thank everyone within Kaiser Permanente and especially within the Web Portal Services Group for your unfaltering support of my completing this dissertation. I want to thank my manager, Mary-Anna Rae, for being so understanding, patient, and supportive. I want to thank everyone for graciously giving up their time to participate in the final user study, including Dani Tobler, Mark Randall, Arturo Delgadillo, Kayvan Sootodeh, Royce Everone, Gaston Cangiano,

Bryan Merrill, Lisa Washington, Katrina Rosario, Anthony Harris, Mary-Anna Rae, Santhosh Pillai, Cheryl Rucks, Kelly Albin, and Barbara Andrews. I want to thank Ron Dolin for bringing me into this special group, and the upper management team for making it possible for me to remain a part of this group – Doug Shelton, Ric Leopold, Kayvan Sotoodeh, Ben Say, and Dr. Henry Neidermeier. I also want to thank the many clients that I have worked with; these assignments helped me to hone my skills as a usability engineer and to put my money where my mouth is.

I want to thank Reggie Walker for helping me to get into graduate school in the first place. I want to acknowledge the family members and friends who played a role in my successfully completing this program: my brother LaVance and his family, my niece Alisia Ivory, Ronald Ringo and family, Damesha Ringo, Betrand Gaskin, Teddy and Teresa Gaskin, Lin Covington, Janice Sherman, Carmen Anthony, Richard Freeman, Dr. Carla Trujillo, Dr. Johnathan Reason, Dr. Adrian Isles, Dr. John Davis, Roy Sutton, and other members of BGESS. I want to acknowledge the faculty and staff at the Head-Royce school for making it possible for my son to receive a quality education. If I have left anyone out, I do apologize. Please know that you are certainly not forgotten. I am just all thought out right now.

Finally, yet importantly, I want to acknowledge the funding sources that made it possible to complete this program: a Microsoft Research Grant, a Gates Millennium Fellowship, a GAANN fellowship, a Lucent Cooperative Research Fellowship Program grant, an AT&T Cooperative Research Fellowship Program grant, an NSF Honorable Mention grant, and others. I want to thank John Choppy in the Berkeley financial aid office for helping me to navigate the financial aid muddle over the years. I want to acknowledge Professor Charles Thompson from the AT&T and Lucent Cooperative Research Fellowship programs for challenging me to finish my program. When can I expect my Montblanc pen?

Chapter 1

Introduction

Despite the abundance of design recommendations, recipes, and guidelines for building a usable Web site [Flanders and Willis 1998; Fleming 1998; Nielsen 1998c; Nielsen 1999b; Nielsen 2000; Rosenfeld and Morville 1998; Sano 1996; Schriver 1997; Shedroff 1999; Shneiderman 1997; Spool *et al.* 1999], usability, especially for information-centric Web sites, continues to be a pressing problem. Given that an estimated 90% of sites provide inadequate usability [Forrester Research 1999], steady growth in new sites [Internet Software Consortium 2001], and a severe shortage of user interface professionals to ensure usable sites [Nielsen 1999b], tools and methodologies are needed to accelerate and improve the Web site design process.

One way to ensure the usability of Web sites is via formal testing with users. Nielsen [1998a] claims that it takes 39 hours to usability test a Web site the first time, including planning the test, defining the test tasks, recruiting test participants, conducting a test with five participants, analyzing the results, and writing the report; with experience, this time can be reduced to 16 hours. Nielsen further claims that a usability test with five participants will typically reveal 80% of the site-level usability problems (e.g., home page, information architecture, navigation and search, linking strategy, etc.) and 50% of the page-level problems (e.g., understandability of headings, links, and graphics). The author advocates increasing page-level usability through other methods such as heuristic evaluation. Contrary to these findings, Spool and Schroeder [2001] have shown that five participants only find 35% of usability problems when the participants do not complete the same tasks. Thus, it appears that usability testing may not be a viable method for accelerating and improving the Web design process.

As a complement to usability testing, many detailed usability guidelines have been developed for both general user interfaces [Open Software Foundation 1991; Smith and Mosier 1986] and for Web page design [Comber 1995; Lynch and Horton 1999]. However, designers have historically experienced difficulties following design guidelines [Borges *et al.* 1996; de Souza and Bevan 1990; Lowgren and Nordqvist 1992; Smith 1986]. Guidelines are often stated at such a high level that it is unclear how to operationalize them. A typical example can be found in Fleming's book [Fleming 1998], which suggests ten principles of successful navigation design including: be easily learned, remain consistent, provide feedback, provide clear visual messages, and support users' goals and behaviors. Fleming also suggests differentiating design among sites intended for community, learning, information, shopping, identity, and entertainment. Although these goals align well with common sense, they are not justified with empirical evidence and are mute on actual implementation.

Other Web-based guidelines are more straightforward to implement. For example, Nielsen [1996] (updated in 1999 [Nielsen 1999a]) claims that the top ten mistakes of Web site design include using frames, long pages, non-standard link colors, and overly long download times. These are

apparently based on anecdotal observational evidence. Another essay by Nielsen [1997] provides guidelines on how to write for the Web, asserting that since users scan Web pages rather than read them, Web page design should aid scannability by using headlines, using colored text for emphasis, and using 50% less text (less than what is not stated) since it is more difficult to read on the screen than on paper. Although reasonable, guidelines like these are not usually supported with empirical evidence.

Furthermore, there is no general agreement about which Web design guidelines are correct. A survey of 21 Web design guidelines found little consistency among them [Ratner *et al.* 1996]. This might result from the fact that there is a lack of empirical validation for such guidelines. Surprisingly, no studies have derived Web design guidelines directly from Web sites that have been evaluated in some way, such as usability testing or heuristic evaluation. This dissertation presents the first attempt to analyze a large collection of example interfaces to develop statistical models for evaluating the quality of new interfaces (applied specifically to Web sites). Such an automated evaluation approach can potentially accelerate and improve the Web design process.

As background for the methodology and tools presented in this dissertation, Chapter 2 summarizes an extensive survey of usability evaluation methods for Web and graphical interfaces. It shows that automated methods are greatly underexplored in the Web domain and that existing methods require some form of usability testing to employ.

It is natural to question what existing and new automated approaches evaluate and furthermore, do they improve interface design. Chapter 2 discusses evaluation methods under the assumption that they all support usability evaluation to some degree. According to ISO9241, usability is the extent to which users can use a computer system to achieve specified goals effectively and efficiently while promoting feelings of satisfaction in a given context of use [International Standards Organization 1999]. Only methods that solicit user input, such as usability testing and surveys, enable the assessment of whether a site is usable according to this definition; thus, other methods (non-automated and automated) may not actually evaluate usability. However, what these methods may evaluate is conformance to usability principles, predicted usability, and possibly other aspects related to the usability of the interface; all of these aspects are important and can potentially increase the likelihood that an interface will be usable.

As discussed above, usability testing may require considerable time, effort, and money and may not reveal all of the problems with a site. Chapter 3 proposes new methods of automated usability evaluation based on measurement, analytical modeling, and simulation methods used in the performance evaluation domain, in particular for evaluating the performance of computer systems. The outcome of these methods is typically quantitative data that can be used to objectively compare systems.

The automated evaluation methodology developed in this dissertation is a synthesis of both performance evaluation and usability evaluation. In particular, the methodology consists of computing an extensive set of quantitative page-level and site-level measures for sites that have been rated by Internet professionals. These measures in conjunction with the expert ratings are used to derive statistical models of highly-rated Web interfaces. As is done with guideline review methods in the usability evaluation domain, the models are then used in the automated analysis of Web pages and sites. However, unlike other guideline review methods, the guidelines in this case are in essence derived from empirical data.

Chapter 4 summarizes the methodology and tools. Chapter 5 describes the page-level and site-level quantitative measures developed as the result of an extensive survey of the Web design literature, including texts written by experts and usability studies. The survey revealed a set of usability aspects, and Chapter 5 describes a total of 141 page-level and 16 site-level measures for assessing many of these aspects, such as the amount of text on a page, the number of colors used,

the download speed, and the consistency of pages in the site.

Chapter 6 describes the analysis of data from over 5300 pages and 330 sites to develop several statistical models for evaluating Web page and site quality. The models distinguish pages and sites that are rated as good, average, and poor by experts and make it possible to take into consideration the context (e.g., the page's functional style or the site's topical category) in which pages and sites are designed. These models include one to assess whether a page is a good home page and one to assess whether a site is a good health information site. The accuracy of page-level models range from 93%–96%, while the accuracy of site-level models range from 68%–88%; the site-level accuracy is considerably less, possibly due to inadequate data.

The methodology developed in this dissertation differs from other automated evaluation methods for Web interfaces in a major way – it is based on empirical data. In essence, the statistical models can be viewed as a reverse engineering of design decisions that went into producing highly-rated interfaces; presumably, these design decisions were informed by user input (e.g., usability testing or surveys). Nonetheless, two studies were conducted to provide insight about the questions posed above (what is evaluated and are designs improved). Chapter 7 describes a study that focused on linking expert ratings to usability ratings. Although the results suggest some relationship between expert and end user ratings, strong conclusions could not be drawn from the study due to problems with the study design.

Chapter 8 demonstrates use of the statistical models for assessing and improving an example site and shows that the model output does inform design improvements. It also provides more insight into what the profiles actually represent and the type of design changes informed by them. Chapter 9 presents findings from a study of this site and four others similarly modified based on model output; site modifications were made by the author and three students – two undergraduates and one graduate. The study focused on determining whether the statistical models help to improve Web designs. Thirteen participants (four professional Web designers, three non-professional designers who had built Web sites, and six participants who had no experience building Web sites) completed two types of tasks during this study: 1. explore alternative versions of Web pages and select the ones exhibiting the highest quality; and 2. explore pages from sites and rate the quality of the site. The results show that participants preferred pages modified based on the Web interface profiles over the original versions, and participants rated modified sites (including the example site) higher than the original sites; the differences were significant in both cases.

Chapter 10 demonstrates use of the statistical models for exploring existing Web design guidelines. The chapter examines contradictory or vague guidelines for nine aspects of Web interfaces, including the amount of text, font styles and sizes, colors, and consistency. The statistical models reveal quantitative thresholds that validate and in some cases invalidate advice in the literature. This examination shows that the methodology makes it possible to derive Web design guidelines directly from empirical data.

Chapter 2

Usability Evaluation of User Interfaces

2.1 Introduction

Usability is the extent to which users can use a computer system to achieve specified goals effectively and efficiently while promoting feelings of satisfaction in a given context of use.[1] Usability evaluation (UE) consists of methodologies for measuring the usability aspects of a system's user interface (UI) and identifying specific problems [Dix *et al.* 1998; Nielsen 1993]. Usability evaluation is an important part of the overall user interface design process, which ideally consists of iterative cycles of designing, prototyping, and evaluating [Dix *et al.* 1998; Nielsen 1993]. Usability evaluation is itself a process that entails many activities depending on the method employed. Common activities include:

- **Capture -** collecting usability data, such as task completion time, errors, guideline violations, and subjective ratings;
- **Analysis -** interpreting usability data to identify problems in the interface; and
- **Critique -** suggesting solutions or improvements to mitigate problems.

A wide range of usability evaluation techniques have been proposed, and a subset of these are currently in common use. Some evaluation techniques, such as formal usability 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 usability problems.

Usability 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; Jeffries *et al.* 1991; Molich *et al.* 1998; Molich *et al.* 1999; Nielsen 1993]. 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 for evaluations of two user interfaces[2]. This result implies a lack of systematicity or predictability in the findings

[1] Adapted from ISO9241 (*Ergonomic requirements for office work with visual display terminals* [International Standards Organization 1999]).

[2] The first study involved four professional teams, while the second study involved seven professional teams and two student teams. Details were not provided about study participants.

of usability evaluations. Furthermore, usability evaluation typically only covers a subset of the possible actions users might take. For these reasons, usability experts often recommend using several different evaluation techniques [Dix *et al.* 1998; Nielsen 1993].

How can systematicity of results and fuller coverage in usability assessment be achieved? One solution is to increase the number of usability teams evaluating the system, and to increase the number of study participants. An alternative is to *automate* some aspects of usability evaluation, such as the capture, analysis, or critique activities.

Automating some aspects of usability evaluation has 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, such as immediate task cancellation.

- 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, usability evaluations 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 usability evaluation. Methods that automate capture, analysis, or critique activities can decrease the time spent on usability 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 usability 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.

1. Specify usability evaluation goals.
2. Determine UI aspects to evaluate.
3. Identify target users.
4. Select usability metrics.
5. Select evaluation method(s).
6. Select tasks.
7. Design experiments.
8. Capture usability data.
9. Analyze and interpret usability data.
10. Critique UI to suggest improvements.
11. Iterate the process if necessary.
12. Present results.

Figure 2.1: Activities that may occur during the usability evaluation process.

Despite the potential advantages, the space of usability evaluation automation is quite underexplored. This chapter presents a detailed survey of UE methods, with an emphasis on automation; a shorter version of this survey is scheduled for publication [Ivory and Hearst 2001]. The chapter begins with a brief overview of the usability evaluation process. It introduces a taxonomy for classifying UE automation and summarizes the application of this taxonomy to 133 usability evaluation methods. Several sections describe these methods in more detail, including summative assessments of automation techniques. The results of this survey suggest promising ways to expand existing approaches to better support usability evaluation; these approaches are also discussed.

2.2 The Usability Evaluation Process

Usability evaluation is a process that entails some of the activities depicted in Figure 2.1, depending on the method used. This section discusses each of these activities. Several literature sources informed this discussion, including [Dix *et al.* 1998; Nielsen 1993; Shneiderman 1998].

2.2.1 Specify Usability Evaluation Goals

Usability evaluation is applicable at all stages of a UI life cycle (e.g., design, implementation, and re-design). At these various stages, different UE goals are relevant. Below is a list of typical UE goals.

- Specify UI requirements
- Evaluate design alternatives
- Identify specific usability problems

- Improve UI performance

The evaluator must clearly specify the goals of the usability evaluation at the outset of the study. These goals influence other aspects of UI assessment, such as the UI components to evaluate and appropriate evaluation methods.

2.2.2 Determine UI Aspects to Evaluate

Some UIs can be extremely large and complex, and an evaluation of all aspects may not be economically feasible. Hence, the evaluator must determine specific UI aspects to evaluate. These aspects must be consistent with the goals of the usability evaluation.

2.2.3 Identify Target Users

An interface may be intended for a large user community, but it is important to determine user characteristics most relevant for the study and for the UI aspects in particular. If users are employed during the study, they need to be as representative of the larger user community as possible.

2.2.4 Select Usability Metrics

Usability metrics are a crucial component of the usability evaluation. The goal in selecting these metrics is to choose a minimal number of metrics that reveal the maximum amount of usability detail for the UI under study. ISO Standard 9241 [International Standards Organization 1999] recommends using effectiveness, efficiency, and satisfaction measures as described below.

- Effectiveness is the accuracy and completeness with which users achieve specified goals. Example metrics include: percentage of goals achieved, functions learned, and errors corrected successfully.
- Efficiency assesses the resources expended in relation to the accuracy and completeness with which users achieve goals. Example metrics include: the time to complete a task, learning time, and time spent correcting errors.
- Satisfaction reflects users' freedom from discomfort and positive attitudes about use of an interface. Example metrics include: ratings for satisfaction, ease of learning, and error handling.

Metrics discussed above are quantitative in nature. Non-quantitative metrics could include, for example, specific heuristic violations identified during a usability inspection.

2.2.5 Select Evaluation Method(s)

Choosing one or more usability evaluation methods is an important step of the UE process. There are five classes of UE methods: usability testing, inspection, inquiry, analytical modeling, and simulation. An inspection entails an evaluator using a set of criteria to identify potential usability problems in an interface, while testing involves an evaluator observing[3] participants interacting with

[3]During some usability testing, such as remote testing, the evaluator may not actually observe a participant interacting with an interface. Other techniques, such as logging (discussed in Section 2.5.2), may be employed to record the interaction for subsequent analysis.

an interface (i.e., completing tasks) to determine usability problems. Similar to usability testing, inquiry methods entail gathering subjective input (e.g., preferences) from participants, typically through interviews, surveys, questionnaires, or focus groups. Analytical modeling and simulation are engineering approaches to UE that enable evaluators to predict usability with user and interface models. Sections 2.5 – 2.9 discuss the five method classes as well as methods within each of the classes in more detail.

UE methods differ along many dimensions, such as resource requirements, costs, results, and applicability (i.e., at what stages of the interface development process). There is a wide range of methods that one could employ at all stages of system development, which actually complicates choosing an appropriate method. Human Factors Engineering, a company specializing in usability evaluation, has an online resource, Ask Usability Advisor [Human Factors Engineering 1999a], that recommends UE methods based on the following usability requirements: software development stage (requirement, design, code, test, and deployment), personnel availability (usability experts, participants, and software developers), usability dimensions to be measured (effectiveness, efficiency, and satisfaction), the need to obtain quantitative measures, and the need to do remote evaluation. There are also two comprehensive archives of UE methods online - Usability Evaluation Methods [Human Factors Engineering 1999b] and the Usability Methods Toolbox [Hom 1998].

UE methods uncover different types of usability problems; therefore, it is often recommended for evaluators to use multiple assessment methods [Jeffries *et al.* 1991; Molich *et al.* 1998; Molich *et al.* 1999; Nielsen 1993]. For example, during a usability test, participants may also complete questionnaires to provide subjective input; thus, enabling evaluators to gather quantitative and qualitative data.

2.2.6 Select Tasks

Tasks are the most crucial part of the usability evaluation [Dix *et al.* 1998; Nielsen 1993; Shneiderman 1998]. They must be appropriate for the UI aspects under study, the target users, and the evaluation method. Other constraints may affect the selection of tasks, such as cost and time limits during usability testing sessions, for instance.

2.2.7 Design Experiments

After completing the previously discussed activities, the evaluator may need to design experiments for collecting usability data. In particular, the evaluator needs to decide on the number of participants (evaluators and users), the evaluation procedure (this is largely dictated by the UE method) as well as on the environment and system setup. The nature of experiments depends on the evaluation method. Experiments may entail: completing tasks in a controlled manner (usability testing); responding to specific questions (inquiry); or comparing alternative designs (analytical modeling and simulation). It is also recommended that the evaluator conduct pilot runs during this phase [Nielsen 1993], especially if user involvement is required.

2.2.8 Capture Usability Data

During this phase, the evaluator employs the UE method to record previously specified usability metrics. For some methods, such as usability testing and inspection, the evaluator may also record specific usability problems encountered during evaluation.

2.2.9 Analyze and Interpret Data

The primary goal of usability data analysis is to summarize the results in a manner that informs interpretation. This summarization may entail statistical techniques based on the goals of the UE. It may also entail creating a list of specific usability problems found along with their severity.

Actually interpreting the results of the study is a key part of the evaluation. It entails using the analysis of usability data to draw conclusions as dictated by the evaluation goals. For example, it may mean concluding that one design is better than another or whether usability requirements have been met.

2.2.10 Critique UI to Suggest Improvements

Ideally, analysis and interpretation of usability data illustrate flaws in the UI design as well as ways to possibly improve the design. Subsequent analysis may be required to verify that suggested improvements actually improve interface usability.

2.2.11 Iterate Process

Analysis and interpretation of usability data may illustrate the need to repeat the UE process. This iteration may be warranted due to the identification of other UI aspects that need evaluation or changes to the UI. Hence, UE may consist of several cycles through this process. This is as expected when an evaluator follows usability engineering or iterative design processes [Dix *et al.* 1998; Nielsen 1993].

2.2.12 Present Results

The final step of the usability evaluation process is to communicate the results and interpretation of these results to the stakeholders. Ideally, the evaluator presents the results such that they can be easily understood (e.g., using graphs and providing severity ratings) and acted upon.

2.3 Taxonomy of Usability Evaluation Automation

In this discussion, a distinction is made between WIMP (Windows, Icons, Pointer, and Mouse) interfaces and Web interfaces, in part because the nature of these interfaces differ and in part because the usability evaluation methods discussed have often only been applied to one type or the other in the literature. WIMP interfaces tend to be more functionally-oriented than Web interfaces. In WIMP interfaces, users complete tasks, such as opening or saving a file, by following specific sequences of operations. Although there are some functional Web applications, most Web interfaces offer limited functionality (i.e., selecting links or completing forms), but the primary role of many Web sites is to provide information. Of course, the two types of interfaces share many characteristics; their differences are highlighted when relevant to usability evaluation.

Several surveys of UE methods for WIMP interfaces exist; Hom [1998] and Human Factors Engineering [1999b] provide a detailed discussion of inspection, inquiry, and testing methods (these terms are defined below). 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, also defined below) and experimental (e.g., usability testing) techniques [Coutaz 1995]. Whitefield *et al.* [1991] present another classification scheme based on the presence

or absence of a user and a computer. Neither of these taxonomies reflect the automated aspects of UE methods.

The sole existing survey of usability evaluation automation, by Balbo [1995], uses a taxonomy which distinguishes among four approaches to automation:

- **Non Automatic**: methods "performed by human factors specialists."
- **Automatic Capture**: methods that "rely on software facilities to record relevant information about the user and the system, such as visual data, speech acts, and keyboard and mouse actions."
- **Automatic Analysis**: methods that are "able to identify usability problems automatically."
- **Automatic Critic**: methods which "not only point out difficulties but propose improvements."

Balbo uses these categories to classify thirteen common and uncommon UE methods. However, most of the methods surveyed require extensive human effort, because they rely on formal usability testing and/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 UI model as input. Thus, this classification scheme is somewhat misleading since it ignores the non-automated requirements of the UE methods.

2.3.1 Proposed Taxonomy

To facilitate discussion of usability evaluation methods, UE methods are grouped along the following four dimensions:

- **Method Class:** describes the type of evaluation conducted at a high level (e.g., usability testing or simulation);
- **Method Type:** describes *how* the evaluation is conducted within a method class, such as thinking-aloud protocol (usability testing class) or information processor modeling (simulation class);
- **Automation Type:** describes the evaluation aspect that is automated (e.g., capture, analysis, or critique); and
- **Effort Level:** describes the type of effort required to execute the method (e.g., model development or interface usage).

Method Class

UE methods are classified into five method classes: testing, inspection, inquiry, analytical modeling, and simulation.

- **Testing:** an evaluator observes participants 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:** participants 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 mimic 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 formative (i.e., identifying specific usability problems) and summative (i.e., obtaining general assessments of usability) purposes. 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, analytical modeling and simulation, are quite similar to performance evaluation techniques used to analyze the performance of computer systems [Jain 1991]. Chapter 3 discusses performance evaluation techniques in detail.

Method Type

There are a wide range of evaluation methods within the testing, inspection, inquiry, analytical modeling, and simulation classes. Rather than discuss each method individually, one or more related methods are grouped into method types; this type typically describes *how* an evaluation is performed. Sections 2.5 – 2.9 present method types.

Automation Type

Balbo's automation taxonomy (described above) was adapted to specify which aspect of a usability evaluation method is automated: none, capture, analysis or critique.

- **None:** no level of automation supported (i.e., evaluator performs all aspects of the evaluation method).
- **Capture:** software automatically records usability data (e.g., logging interface usage).
- **Analysis:** software automatically identifies potential usability problems.
- **Critique:** software automates analysis and suggests improvements.

Effort Level

Balbo's automation taxonomy is also expanded to include consideration of a method's non-automated requirements. Each UE method is augmented with an attribute called **effort level**; this indicates the human effort required for method execution:

- **Minimal Effort:** does not require interface usage or modeling.
- **Model Development:** requires the evaluator to develop a UI model and/or a user model to employ the method.

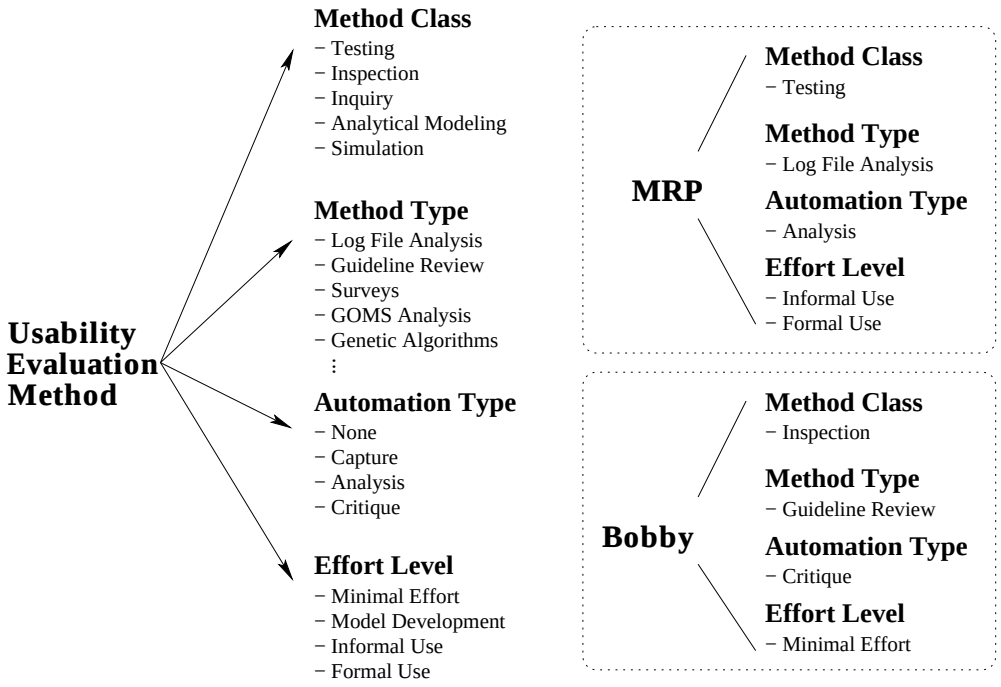


Figure 2.2: Summary of the proposed taxonomy for classifying usability evaluation methods. The right side of the figure demonstrates the taxonomy with two evaluation methods that will be discussed in later sections.

- **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).

These levels are not necessarily ordered by the amount of effort required, since this depends on the method used.

Summary

Figure 2.2 provides a synopsis of the proposed taxonomy and demonstrates it with two evaluation methods. The taxonomy consists of: a method class (testing, inspection, inquiry, analytical modeling, and simulation); a method type (e.g., log file analysis, guideline review, surveys, etc.); an automation type (none, capture, analysis, and critique); and an effort level (minimal, model, informal, and formal). This taxonomy is used to analyze evaluation methods in the remainder of this chapter.

2.4 Overview of Usability Evaluation Methods

Seventy-five UE methods applied to WIMP interfaces and fifty-eight methods applied to Web UIs were surveyed and analyzed using the proposed taxonomy. Of these 133 methods, only 29 apply to both Web and WIMP UIs. The applicability of each method was determined based on

the types of interfaces a method was used to evaluate in the literature and the author's judgment of whether or not the method could be used with other types of interfaces. Tables 2.1 and 2.2 combine survey results for both types of interfaces showing method classes (bold entries in the first column) and method types within each class (entries that are not bold in the first column). Each entry in columns two through five depicts specific UE methods along with the automation support available and the human effort required to employ automation. For some UE methods, more than one approach will be discussed; hence, the number of methods surveyed is shown in parenthesis beside the effort level.

Some approaches provide automation support for multiple method types (see Appendix A). Hence, Tables 2.1 and 2.2 contain only 111 methods. Tables 2.1 and 2.2 also depict methods applicable to both WIMP and Web UIs only once. Table 2.3 provides descriptions of all method types.

There are major differences in automation support among the five method classes. Overall, automation patterns are similar for WIMP and Web interfaces, with the exception that analytical modeling and simulation are far less explored in the Web domain than for WIMP interfaces (two vs. sixteen methods). Appendix A shows the information in Tables 2.1 and 2.2 separated by UI type. Table 2.4 summarizes the number of non-automated and automated capture, analysis, and critique methods surveyed overall and for each type of interface.

Tables 2.1 and 2.2 show that automation in general is greatly underexplored. Table 2.4 shows that methods without automation support represent 64% of the methods surveyed, while methods with automation support collectively represent only 36%. Of this 36%, capture methods represent 15%, analysis methods represent 19% and critique methods represent 2%. All but two of the capture methods require some level of interface usage; genetic algorithms and information scent modeling both use simulation to generate usage data for subsequent analysis. Of all of the surveyed methods, only 29% are free from requirements of formal or informal interface use.

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 one method type: guideline review (e.g., [Farenc and Palanque 1999; Lowgren and Nordqvist 1992; Scholtz and Laskowski 1998]). Guideline review methods automatically detect and report usability violations and then make suggestions for fixing them (discussed further in Section 2.6).

Of those methods that support the next level of automation – analysis – Tables 2.1 and 2.2 show that analytical modeling and simulation methods represent the majority. Most of these methods do not require formal or informal interface use.

The next sections discuss the various UE methods and their automation in more detail. Some methods are applicable to both WIMP and Web interfaces; however, distinctions are made where necessary about a method's applicability. The discussion also presents assessments of automated capture, analysis, and critique techniques using the following criteria:

- **Effectiveness:** how well a method discovers usability problems,
- **Ease of use:** how easy is a method to employ,
- **Ease of learning:** how easy is a method to learn, and
- **Applicability:** how widely applicable is a method to WIMP and/or Web UIs other than those originally applied to.

The effectiveness, ease of use, ease of learning, and applicability of automated methods is highlighted in discussions of each method class.

Method Class / Method Type	**Automation Type**			
	None	**Capture**	**Analysis**	**Critique**
Testing				
Thinking-aloud Protocol	F (1)			
Question-asking Protocol	F (1)			
Shadowing Method	F (1)			
Coaching Method	F (1)			
Teaching Method	F (1)			
Co-discovery Learning	F (1)			
Performance Measurement	F (1)	F (8)		
Log File Analysis			IFM (20)*	
Retrospective Testing	F (1)			
Remote Testing		IF (3)		
Inspection				
Guideline Review	IF (6)		(8)	M (11)†
Cognitive Walkthrough	IF (2)	F (1)		
Pluralistic Walkthrough	IF (1)			
Heuristic Evaluation	IF (1)			
Perspective-based Inspection	IF (1)			
Feature Inspection	IF (1)			
Formal Usability Inspection	F (1)			
Consistency Inspection	IF (1)			
Standards Inspection	IF (1)			
Inquiry				
Contextual Inquiry	IF (1)			
Field Observation	IF (1)			
Focus Groups	IF (1)			
Interviews	IF (1)			
Surveys	IF (1)			
Questionnaires	IF (1)	IF (2)		
Self-reporting Logs	IF (1)			
Screen Snapshots	IF (1)			
User Feedback	IF (1)			

Table 2.1: Automation support for WIMP and Web UE methods (Table 1 of 2). 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). The * for the IFM entry indicates that either formal or informal interface use is required. In addition, a model may be used in the analysis. The † indicates that methods may or may not require a model.

Method Class	Automation Type			
Method Type	None	Capture	Analsis	Critique
Analytical Modeling				
GOMS Analysis	M (4)		M (2)	
UIDE Analysis			M (2)	
Cognitive Task Analysis			M (1)	
Task-environment Analysis	M (1)			
Knowledge Analysis	M (2)			
Design Analysis	M (2)			
Programmable User Models			M (1)	
Simulation				
Information Proc. Modeling			M (9)	
Petri Net Modeling			FM (1)	
Genetic Algorithm Modeling		(1)		
Information Scent Modeling		M (1)		

Table 2.2: Automation support for WIMP and Web UE methods (Table 2 of 2). 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).

2.5 Usability Testing Methods

Usability testing with real participants is a fundamental evaluation method [Nielsen 1993; Shneiderman 1998]. It provides an evaluator with direct information about how people use computers and what some of the 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 or software records the results of the participants' work. The tester then uses these results to determine how well the interface supports users' task completion and to derive other measures, such as the number of errors and task completion time.

Automation has been used predominantly in two ways within usability testing: automated capture of use data and automated analysis of this data according to some metrics or a model (referred to as log file analysis in Table 2.1). In rare cases methods support both automated capture and analysis of usage data [Al-Qaimari and McRostie 1999; Hong *et al.* 2001; Uehling and Wolf 1995].

2.5.1 Usability Testing Methods: Non-automated

This section provides a synopsis of eight non-automated method types. All method types have been or could be applied to both WIMP and Web UIs and require formal interface usage. Two testing protocols, thinking-aloud and question-asking, and six non-automated method types were surveyed; the testing protocols can be used with the other six method types in most cases. The major differences among the testing method types are the actual testing procedure (e.g., participants are silent, think aloud, or have the ability to ask an expert questions) and the intended outcome of testing (e.g., an understanding of the participant's mental model of the system or quantitative data).

Method types are summarized below. The method type (i.e., how an evaluation is performed) and the method (i.e., specific instantiation of a method type) are the same in all cases. Unless otherwise noted, most discussions are based on [Dix *et al.* 1998; Hom 1998; Human Factors Engineering 1999b; Nielsen 1993; Shneiderman 1998].

Method Class Method Type	**Description**
Testing	
Thinking-aloud Protocol	user talks during test
Question-asking Protocol	tester asks user questions
Shadowing Method	expert explains user actions to tester
Coaching Method	user can ask an expert questions
Teaching Method	expert user teaches novice user
Co-discovery Learning	two users collaborate
Performance Measurement	tester or software records usage data during test
Log File Analysis	tester analyzes usage data
Retrospective Testing	tester reviews videotape with user
Remote Testing	tester and user are not co-located during test
Inspection	
Guideline Review	expert checks guideline conformance
Cognitive Walkthrough	expert simulates user's problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies heuristic violations
Perspective-based Inspection	expert conducts narrowly focused heuristic evaluation
Feature Inspection	expert evaluates product features
Formal Usability Inspection	experts conduct formal heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance
Inquiry	
Contextual Inquiry	interviewer questions users in their environment
Field Observation	interviewer observes system use in user's environment
Focus Groups	multiple users participate in a discussion session
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaires	user provides answers to specific questions
Self-reporting Logs	user records UI operations
Screen Snapshots	user captures UI screens
User Feedback	user submits comments
Analytical Modeling	
GOMS Analysis	predict execution and learning time
UIDE Analysis	conduct GOMS analysis within a UIDE
Cognitive Task Analysis	predict usability problems
Task-environment Analysis	assess mapping of user's goals into UI tasks
Knowledge Analysis	predict learnability
Design Analysis	assess design complexity
Programmable User Models	write program that acts like a user
Simulation	
Information Proc. Modeling	mimic user interaction
Petri Net Modeling	mimic user interaction from usage data
Genetic Algorithm Modeling	mimic novice user interaction
Information Scent Modeling	mimic Web site navigation

Table 2.3: Descriptions of the WIMP and Web UE method types depicted in Table 2.1.

Methods Surveyed	Automation Type			
	None	Capture	Analysis	Critique
Overall				
Total	30	7	9	1
Percent	64%	15%	19%	2%
WIMP UIs				
Total	30	5	8	1
Percent	68%	11%	18%	2%
Web UIs				
Total	26	5	4	1
Percent	72%	14%	11%	3%

Table 2.4: Summary of UE methods surveyed for each automation type.

Thinking-aloud Protocol. The thinking-aloud protocol requires participants to verbalize their thoughts, feelings, and opinions during a usability test. One goal of this approach is to enable the tester to get a better understanding of the participant's mental model during interaction with the interface. Critical response and periodic report are two variations of the protocol wherein the participant is vocal only during the execution of certain pre-determined tasks or at pre-determined intervals of time, respectively.

Question-asking Protocol. This method is an extension of the thinking-aloud protocol wherein testers prompt participants by asking direct questions about the interface. The goal of such questioning is to enable the tester to get an even better understanding of the participant's mental model of the system.

Coaching Method. The coaching method allows participants to ask any system-related questions of an expert coach during usability testing. Usually, the tester acts as the expert coach, but it is possible to have a separate tester serving as a coach. The latter approach may allow the tester to gain additional usability insight through observing the interaction between the participant and coach. In cases where an expert user serves as the coach, this also enables the tester to analyze the expert user's mental model of the system. The main goal of this method is to determine the information needs of users to provide better training and documentation in addition to possibly redesigning the interface to eliminate the need for questions in the first place. It is also possible for the tester to control the answers given to questions during testing to discover what types of answers help users the most.

Teaching Method. For this method, the participant interacts with the system first to develop expertise to subsequently teach a novice user about the system. The novice user serves as a student and does not actively engage in problem solving. The participant does the problem solving, explains to the novice user how the system works, and demonstrates a set of pre-determined tasks. This method enables testers to assess the ease of learning of an interface.

Shadowing Method. Shadowing is an alternative to the thinking-aloud protocol wherein an expert user sits next to the tester and explains the participant's behavior during the testing session. Evaluators use this method in situations where it is inappropriate for participants to think aloud or talk to the tester (e.g., collecting performance measurements).

Co-discovery Learning. During a co-discovery learning session, two participants attempt to perform the tasks together while the tester observes their interaction. This approach is similar

to the type of collaboration that occurs naturally in other environments, such as at work. As the participants complete tasks, the tester encourages them to explain what they are thinking about in a manner similar to the thinking-aloud protocol.

Performance Measurement. The goal of this testing method is to capture quantitative data about participants' performance when they complete tasks. As such, there is usually no interaction between the tester and participant during the test. Evaluators usually conduct such testing in a usability lab to facilitate accurate data collection and to minimize interference. Sometimes this method is combined with other techniques to capture qualitative data as well, such as retrospective testing (discussed immediately below).

Measurement studies form the foundation of usability testing, since evaluators can use the results to assess whether the usability goals have been met as well as for competitive analysis. In the first case the evaluator would re-define an abstract performance goal, such as usability, into a specific usability attribute, such as efficiency of use. After specifying a specific usability attribute, the evaluator can quantify this attribute with a metric (e.g., time to complete a task, time spent recovering from errors, etc.) and devise a plan for measuring this metric in the interface and collecting the necessary performance data. Without automated tools, this collection is typically accomplished by taking notes or video taping testing sessions and subsequently reviewing the videotape to compute performance measures.

MUSiC (Metrics for Usability Standards in Computing) [Bevan and Macleod 1994; Macleod *et al.* 1997] is a rigorous performance measurement method developed by a consortium of European institutions, including Serco Usability Services (formerly the National Physical Laboratory), the University College Cork, and the HUSAT (Human Sciences and Advanced Technology) Institute. Applying the methodology entails: conducting a formal usability context analysis (i.e., determining who the users are, how they use the UI, and in what situations they use it) and following the performance measurement method [Rengger *et al.* 1993] as prescribed. MUSiC includes tools to support automated analysis of video recording using DRUM (Diagnostic Recorder for Usability Measurement, discussed in Section 2.5.2) [Macleod and Rengger 1993] as well as collecting subjective usability data via the SUMI (Software Usability Measurement Inventory, discussed in Section 2.7.2) [Porteous *et al.* 1993] questionnaire.

Retrospective Testing. This method is a followup to any other videotaped testing session wherein the tester and participant review the videotape together. During this review, the tester asks the participant questions regarding her behavior during the test. The goal of this review is to collect additional information from the usability test. Although such testing can be valuable, it substantially increases the cost of usability testing because each test takes at least twice as long to conduct.

2.5.2 Usability Testing Methods: Automated Capture

Many usability testing methods require the recording of actions a user makes while exercising an interface. This can be done by an evaluator taking notes while the participant uses the system, either live or by repeatedly viewing a videotape of the session; both are time-consuming activities. As an alternative, automated capture techniques can log user activity automatically. An important distinction can be made between information that is easy to record but difficult to interpret, such as keystrokes, and information that is meaningful but difficult to automatically

Method Class: Testing		
Automation Type: Capture		
Method Type: Performance Measurement - software records usage data during test (8 methods)		
UE Method	**UI**	**Effort**
Log low-level events ([Hammontree *et al.* 1992])	WIMP	F
Log UIMS events (UsAGE, IDCAT)	WIMP	F
Log system-level events (KALDI)	WIMP	F
Log Web server requests ([Scholtz and Laskowski 1998])	Web	F
Log client-side activities (WebVIP, WET)	Web	F
Log Web proxy requests (WebQuilt)	Web	F
Method Type: Remote Testing - tester and user are not co-located (3 methods)		
UE Method	**UI**	**Effort**
Employ same-time different-place testing (KALDI)	WIMP, Web	IF
Employ different-time different-place testing (journaled sessions)	WIMP, Web	IF
Analyze a Web site's information organization (WebCAT)	Web	IF

Table 2.5: Synopsis of automated capture support for usability testing methods.

label, such as task completion. Automated capture approaches vary with respect to the granularity of information captured.

Within the usability testing class of UE, automated capture of usage data is supported by two method types: performance measurement and remote testing. Both require the instrumentation of a user interface, incorporation into a user interface management system (UIMS), or capture at the system level. A UIMS [Olsen, Jr. 1992] is a software library that provides high-level abstractions for specifying portable and consistent interface models that are then compiled into UI implementations on each platform similarly to Java programs. Table 2.5 provides a synopsis of automated capture methods discussed in the remainder of this section. Support available for WIMP and Web UIs is discussed separately.

Usability Testing Methods: Automated Capture – WIMP UIs

Performance measurement methods record usage data (e.g., a log of events and times when events occurred) during a usability test. Video recording and event logging tools [Al-Qaimari and McRostie 1999; Hammontree *et al.* 1992; Uehling and Wolf 1995] are available to automatically and accurately align timing data with user interface events. Some event logging tools (e.g., [Hammontree *et al.* 1992]) record events at the keystroke or system level. Recording data at this level produces voluminous log files and makes it difficult to map recorded usage into high-level tasks.

As an alternative, two systems log events within a UIMS. UsAGE (User Action Graphing Effort)[4] [Uehling and Wolf 1995] enables the evaluator to replay logged events, meaning it can replicate logged events during playback. This requires that the same study data (databases, documents, etc.) be available during playback as was used during the usability test. IDCAT (Integrated Data Capture and Analysis Tool) [Hammontree *et al.* 1992] logs events and automatically filters and classifies them into meaningful actions. This system requires a video recorder to synchronize taped footage with logged events. KALDI (Keyboard/mouse Action Logger and Display Instrument) [Al-Qaimari and McRostie 1999] supports event logging and screen capturing via Java and

[4]This method is not to be confused with the USAGE analytical modeling approach discussed in Section 2.8.

does not require special equipment. Both KALDI and UsAGE also support log file analysis (see Section 2.5.3).

Remote testing methods enable testing between a tester and participant who are not co-located. In this case the evaluator is not able to observe the user directly, but can gather data about the process over a computer network. Remote testing methods are distinguished according to whether or not a tester observes the participant during testing or not. Same-time different-place and different-time different-place are two major remote testing approaches [Hartson *et al.* 1996].

In same-time different-place or remote-control testing the tester observes the participant's screen through network transmissions (e.g., using PC Anywhere or Timbuktu) and may be able to hear what the participant says via a speaker telephone or a microphone affixed to the computer. Software makes it possible for the tester to interact with the participant during the test, which is essential for techniques like the question-asking or thinking aloud protocols that require such interaction.

The tester does not observe the user during different-time different-place testing. An example of this approach is the journaled session [Nielsen 1993], in which software guides the participant through a testing session and logs the results. Evaluators can use this approach with prototypes to get feedback early in the design process, as well as with released products. In the early stages, evaluators distribute disks containing a prototype of a software product and embedded code for recording users' actions. Users experiment with the prototype and return the disks to evaluators upon completion. It is also possible to embed dialog boxes within the prototype to record users' comments or observations during usage. For released products, evaluators use this method to capture statistics about the frequency with which the user has used a feature or the occurrence of events of interest (e.g., error messages). This information is valuable for optimizing frequently-used features and the overall usability of future releases.

Remote testing approaches allow for wider testing than traditional methods, but evaluators may experience technical difficulties with hardware and/or software components (e.g., inability to correctly configure monitoring software or network failures). This can be especially problematic for same-time different-place testing where the tester needs to observe the participant during testing. Most techniques also have restrictions on the types of UIs to which they can be applied. This is mainly determined by the underlying hardware (e.g., PC Anywhere only operates on PC platforms) [Hartson *et al.* 1996]. KALDI, mentioned above, also supports remote testing. Since it was developed in Java, evaluators can use it for same- and different-time testing of Java applications on a wide range of computing platforms.

Usability Testing Methods: Automated Capture – Web UIs

The Web enables remote testing and performance measurement on a much larger scale than is feasible with WIMP interfaces. Both same-time different-place and different-time different-place approaches can be employed for remote testing of Web UIs. Similar to journaled sessions, Web servers maintain usage logs and automatically generate a log file entry for each request. These entries include the IP address of the requester, request time, name of the requested Web page, and in some cases the URL of the referring page (i.e., where the user came from). Server logs cannot record user interactions that occur only on the client side (e.g., use of within-page anchor links or back button), and the validity of server log data is questionable due to caching by proxy servers and browsers [Etgen and Cantor 1999; Scholtz and Laskowski 1998]. Server logs may not reflect usability, especially since these logs are often difficult to interpret [Schwartz 2000] and users' tasks may not be discernible [Byrne *et al.* 1999; Schwartz 2000].

Client-side logs capture more accurate, comprehensive usage data than server-side logs

because they allow all browser events to be recorded. Such logging may provide more insight about usability. On the downside, it requires every Web page to be modified to log usage data, or else use of an instrumented browser or special proxy server.

The NIST WebMetrics tool suite [Scholtz and Laskowski 1998] captures client-side usage data. This suite includes WebVIP (Web Visual Instrumentor Program), a visual tool that enables the evaluator to add event handling code to Web pages. This code automatically records the page identifier and a time stamp in an ASCII file every time a user selects a link. (This package also includes a visualization tool, VISVIP [Cugini and Scholtz 1999], for viewing logs collected with WebVIP; see Section 2.5.3.) Using this client-side data, the evaluator can accurately measure time spent on tasks or particular pages as well as study use of the back button and user clickstreams. Despite its advantages over server-side logging, WebVIP requires the evaluator to make a copy of an entire Web site, which could lead to invalid path specifications and other difficulties with getting the copied site to function properly. The evaluator must also add logging code to each individual link on a page. Since WebVIP only collects data on selected HTML links, it does not record interactions with other Web objects, such as forms. It also does not record usage of external or non-instrumented links.

Similar to WebVIP, the Web Event-logging Tool (WET) [Etgen and Cantor 1999] supports the capture of client-side data, including clicks on Web objects, window resizing, typing in a form object and form resetting. WET interacts with Microsoft Internet Explorer and Netscape Navigator to record browser event information, including the type of event, a time stamp, and the document-window location. This gives the evaluator a more complete view of the user's interaction with a Web interface than WebVIP. WET does not require as much effort to employ as WebVIP, nor does it suffer from the same limitations. To use this tool, the evaluator specifies events (e.g., clicks, changes, loads, and mouseovers) and event handling functions in a text file on the Web server; sample files are available to simplify this step. The evaluator must also add a single call to the text file within the <head> tag of each Web page to be logged. Currently, the log file analysis for both WebVIP and WET is manual. Future work has been proposed to automate this analysis.

As an alternative to server-side and client-side logging, WebQuilt [Hong *et al.* 2001] uses proxy-based logging to capture usage data. The system automatically captures Web server requests using a special proxy server, logs requests, and subsequently routes requests to the Web server. All of the links in Web pages are also redirected to the proxy server; this eliminates the need for users to manually configure their browsers to route requests to the proxy. The system captures more accurate site usage details (e.g., use of the back button) than server-side logging, makes it possible to run usability tests on any Web site (e.g., for competitive analysis), and makes it possible to track participants accurately, since extra information can be encoded in the study URL. Although the system does not capture many client-side details, such as the use of page elements or window resizing, it does simplify instrumenting a site for logging, since this is done automatically. The WebQuilt system also supports task-based analysis and visualization of captured usage data (see Section 2.5.3).

The NIST WebMetrics tool suite also includes WebCAT (Category Analysis Tool), a tool that aids in Web site category analysis, by a technique sometimes known as card sorting [Nielsen 1993]. In non-automated card sorting, the evaluator (or a team of evaluators) writes concepts on pieces of paper, and users group the topics into piles. The evaluator manually analyzes these groupings to determine a good category structure. WebCAT allows the evaluator to test proposed topic categories for a site via a category matching task (a variation of card-sorting where users assign concepts to predefined categories); this task can be completed remotely by users. Results are compared to the designer's category structure, and the evaluator can use the analysis to inform the best information organization for a site. WebCAT enables wider testing and faster analysis than

traditional card sorting, and helps make the technique scale for a large number of topic categories.

Usability Testing Methods: Automated Capture – Discussion

Automated capture methods represent important first steps toward informing UI improvements – they provide input data for analysis and in the case of remote testing, enable the evaluator to collect data for a larger number of users than traditional methods. Without this automation, evaluators would have to manually record usage data, expend considerable time reviewing videotaped testing sessions or in the case of the Web, rely on questionable server logs. Methods such as KALDI and WET capture high-level events that correspond to specific tasks or UI features. KALDI also supports automated analysis of captured data as discussed below.

Table 2.5 summarizes performance measurement and remote testing methods discussed in this section. It is difficult to assess the ease of use and learning of these approaches, especially IDCAT and remote testing approaches that require integration of hardware and software components, such as video recorders and logging software. For client-side Web site logging, WET appears to be easier to use and learn than WebVIP. It requires the creation of an event handling file and the addition of a small block of code in each Web page header, while WebVIP requires the evaluator to add code to every link on all Web pages. WET also enables the evaluator to capture more comprehensive usage data than WebVIP. WebQuilt is easier to use and learn than WET because a proxy server automatically instruments a site for logging; however, it does not capture the same level of detail as client-side logging. WebCAT appears straightforward to use and learn for topic category analysis. Both remote testing and performance measurement techniques have restrictions on the types of UIs to which they can be applied. This is mainly determined by the underlying hardware (e.g., PC Anywhere only operates on PC platforms) or UIMS, although KALDI can potentially be used to evaluate Java applications on a wide range of platforms.

2.5.3 Usability Testing Methods: Automated Analysis

Log file analysis methods automate the analysis of data captured during formal or informal interface use. Since Web servers automatically log client requests, log file analysis is a heavily used methodology for evaluating Web interfaces [Drott 1998; Fuller and de Graaff 1996; Hochheiser and Shneiderman 2001; Sullivan 1997]. The survey revealed four general approaches for analyzing WIMP and Web log files: metric-based, pattern-matching, task-based, and inferential. Table 2.6 provides a synopsis of automated analysis methods discussed in the remainder of this section. Support available for the four general approaches is discussed separately.

Usability Testing Methods: Automated Analysis – Metric-Based Analysis of Log Files

Metric-based approaches generate quantitative performance measurements. Three examples for WIMP interfaces are DRUM [Macleod and Rengger 1993], the MIKE UIMS [Olsen, Jr. and Halversen 1988], and AMME (Automatic Mental Model Evaluator) [Rauterberg 1995; Rauterberg 1996b; Rauterberg and Aeppili 1995]. DRUM enables the evaluator to review a video tape of a usability test and manually log starting and ending points for tasks. DRUM processes this log and derives several measurements, including: task completion time, user efficiency (i.e., effectiveness divided by task completion time), and productive period (i.e., portion of time the user did not have problems). DRUM also synchronizes the occurrence of events in the log with videotaped footage, thus speeding up video analysis.

The MIKE UIMS enables an evaluator to assess the usability of a UI specified as a model that can be rapidly changed and compiled into a functional UI. MIKE captures usage data and

Method Class: Testing		
Automation Type: Analysis		
Method Type: Log File Analysis - analyze usage data (20 methods)		
UE Method	**UI**	**Effort**
Use metrics during log file analysis (DRUM, MIKE UIMS, AMME)	WIMP	IF
Use metrics during log file analysis (Service Metrics, [Bacheldor 1999])	Web	IF
Use pattern matching during log file analysis (MRP)	WIMP	IF
Use task models during log file analysis (IBOT, QUIP, WebQuilt, KALDI, UsAGE)	WIMP	IF
Use task models and pattern matching during log file analysis (ÉMA, USINE, RemUSINE)	WIMP	IFM
Visualization of log files ([Guzdial *et al.* 1994])	WIMP	IF
Statistical analysis or visualization of log files (traffic- and time-based analyses, VISVIP, Starfield and Dome Tree visualizations)	Web	IF

Table 2.6: Synopsis of automated analysis support for usability testing methods.

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. MIKE also calculates these metrics separately for command selection (e.g., traversing a menu, typing a command name, or hitting a button) and command specification (e.g., entering arguments for a command) to help the evaluator locate specific problems within the UI.

AMME employs petri nets [Petri 1973] to reconstruct and analyze the user's problem solving process. It requires a specially-formatted log file and a manually-created system description file (i.e., a list of interface states and a state transition matrix) to generate the petri net. It then computes measures of behavioral complexity (i.e., steps taken to perform tasks), routinization (i.e., repetitive use of task sequences), and ratios of thinking vs. waiting time. User studies with novices and experts validated these quantitative measures and showed behavioral complexity to correlate negatively with learning (i.e., less steps are taken to solve tasks as a user learns the interface) [Rauterberg and Aeppili 1995]. Hence, the behavioral complexity measure provides insight on interface complexity. It is also possible to simulate the generated petri net (see Section 2.9) to further analyze the user's problem solving and learning processes. Multidimensional scaling and Markov analysis tools are available for comparing multiple petri nets (e.g., nets generated from novice and expert user logs). Since AMME processes log files, it could easily be extended to Web interfaces.

For the Web, site analysis tools developed by Service Metrics [Service Metrics 1999] and others [Bacheldor 1999] allow evaluators to pinpoint performance bottlenecks, such as slow server response time, that may negatively impact the usability of a Web site. Service Metrics' tools, for example, can collect performance measures from multiple geographical locations under various access conditions. In general, performance measurement approaches focus on server and network performance, but provide little insight into the usability of the Web site itself.

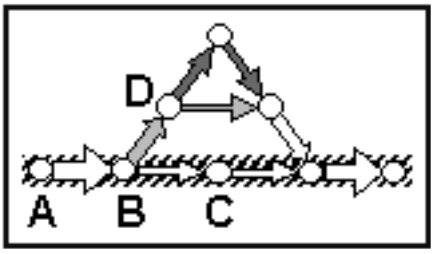


Figure 2.3: QUIP usage profile contrasting task flows for two users to the designer's task flow (diagonal shading) [Helfrich and Landay 1999]. Each node represents a user action, and arrows indicate actions taken by users. The width of arrows denotes the fraction of users completing actions, while the color of arrows reflects the average time between actions (darker colors correspond to longer time). Reprinted with permission of the authors.

Usability Testing Methods: Automated Analysis – Pattern-Matching Analysis of Log Files

Pattern-matching approaches, such as MRP (Maximum Repeating Pattern) [Siochi and Hix 1991], analyze user behavior captured in logs. MRP detects and reports repeated user actions (e.g., consecutive invocations of the same command and errors) that may indicate usability problems. Studies with MRP showed the technique to be useful for detecting problems with expert users, but additional data prefiltering was required for detecting problems with novice users. Whether the evaluator performed this prefiltering or it was automated is unclear in the literature.

Three evaluation methods employ pattern matching in conjunction with task models. These methods are discussed immediately below.

Usability Testing Methods: Automated Analysis – Task-Based Analysis of Log Files

Task-based approaches analyze discrepancies between the designer's anticipation of the user's task model and what a user actually does while using the system. The IBOT system [Zettlemoyer *et al.* 1999] automatically analyzes log files to detect task completion events. The IBOT system interacts with Windows operating systems to capture low-level window events (e.g., keyboard and mouse actions) and screen buffer information (i.e., a screen image that can be processed to automatically identify widgets). The system then combines this information into interface abstractions (e.g., menu select and menubar search operations). Evaluators can use the system to compare user and designer behavior on these tasks and to recognize patterns of inefficient or incorrect behaviors during task completion. Without such a tool, the evaluator has to study the log files and do the comparison manually. Future work has been proposed to provide critique support.

The QUIP (Quantitative User Interface Profiling) tool [Helfrich and Landay 1999] and KALDI [Al-Qaimari and McRostie 1999] (see previous section) provide more advanced approaches to task-based, log file analysis for Java 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-based and trace-based information into directed graphs (see Figure 2.3). For example, the average time between actions is indicated by the color of each arrow, and the proportion of users who performed a particular sequence of actions is indicated by the width of each arrow. The designer's task flow is indicated by the diagonal shading in Figure 2.3. Currently, the evaluator must instrument the UI to collect the necessary usage data, and must manually analyze the graphs to identify usability problems.

WebQuilt [Hong *et al.* 2001] provides a graphical depiction of usage data captured via a

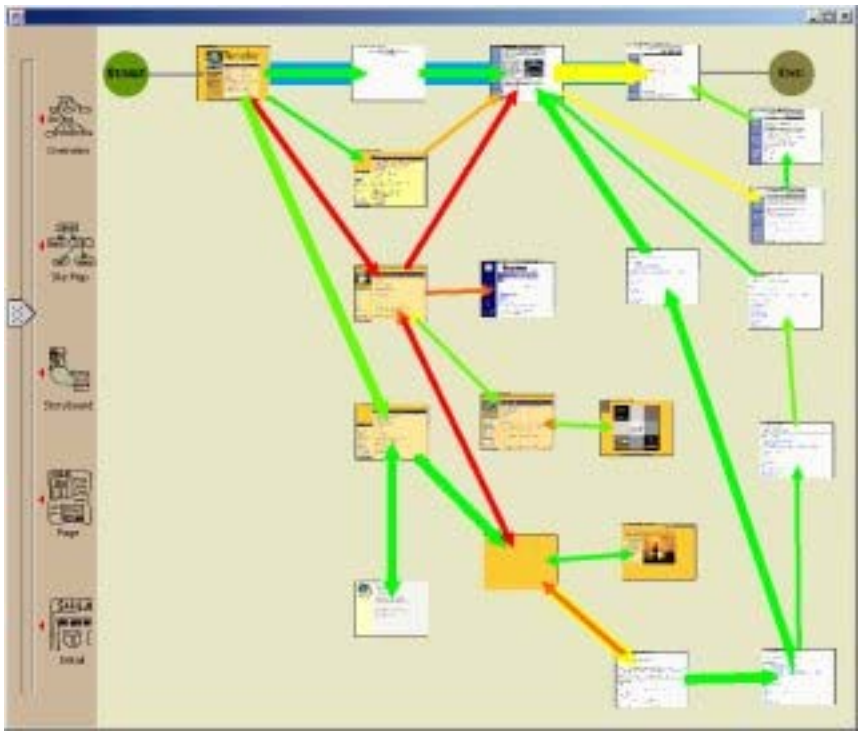


Figure 2.4: WebQuilt visualization contrasting task flows for twelve users to the designer's task flow (path across the top with thick shading) [Hong *et al.* 2001]. The circle on the left shows the start of the task, while the circle on the right shows the end of the task. Each thumbnail corresponds to a page in the Web site, and arrows indicate actions taken by users. The width of arrows denotes the fraction of users traversing the path, while the color of arrows reflects the average time users spent on pages before clicking a link (darker colors correspond to longer time). The interface enables evaluators to view usage data at different levels of detail. Reprinted with permission of the authors.

special proxy server. The visualization is very similar to QUIP, in that it aggregates usage data and summarizes task completion with arrows showing actions taken by users, the percentages of users taking the actions (width), and the average times users spent on pages before selecting links (color). WebQuilt also enables the evaluator to view the usage data at multiple levels of detail using a zooming interface. For example, if users spent a considerable amount of time on a Web page, the evaluator could view the actual page within the interface.

KALDI captures usage data and screen shots for Java applications. It also enables the evaluator to classify tasks (both manually and via automatic filters), compare user performance on tasks, and playback synchronized screen shots. It depicts logs graphically to facilitate analysis.

UsAGE [Uehling and Wolf 1995], which also supports logging usage data within a UIMS, provides a similar graphical presentation for comparing event logs for expert and novice users. However, graph nodes are labeled with UIMS event names, thus making it difficult to map events to specific interface tasks. To mitigate this shortcoming, UsAGE allows the evaluator to replay recorded events in the interface.

Several systems incorporate pattern-matching (see discussion above) into their analyses. This combination results in the most advanced log file analysis of all of the approaches surveyed. These systems include ÉMA (Automatic Analysis Mechanism for the Ergonomic Evaluation of User Interfaces) [Balbo 1996], USINE (USer Interface Evaluator) [Lecerof and Paternò 1998], and RemUSINE (Remote USer Interface Evaluator) [Paternò and Ballardin 1999], all discussed below.

ÉMA uses a manually-created data-flow task model and standard behavior heuristics to flag usage patterns that may indicate usability problems. ÉMA extends the MRP approach (repeated command execution) to detect additional patterns, including immediate task cancellation, shifts in direction during task completion, and discrepancies between task completion and the task model. ÉMA outputs results in an annotated log file, which the evaluator must manually inspect to identify usability problems. Application of this technique to the evaluation of ATM (Automated Teller Machine) usage corresponded with problems identified using standard heuristic evaluation [Balbo 1996].

USINE [Lecerof and Paternò 1998] employs the ConcurTaskTrees [Paternò *et al.* 1997] notation to express temporal relationships among UI tasks (e.g., enabling, disabling, and synchronization). Using this 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 reflected in the usage data. Studies with a graphical interface showed that USINE's results correspond with empirical observations and highlight the source of some usability problems. To use the system, evaluators must create task models using the ConcurTaskTrees editor as well as a table specifying mappings between log entries and the task model. USINE processes log files and outputs detailed reports and graphs to highlight usability problems. RemUSINE [Paternò and Ballardin 1999] is an extension that analyzes multiple log files (typically captured remotely) to enable comparison across users.

Usability Testing Methods: Automated Analysis – Inferential Analysis of Log Files

Inferential analysis of Web log files includes both statistical and visualization techniques. Statistical approaches include traffic-based analysis (e.g., pages-per-visitor or visitors-per-page) and time-based analysis (e.g., click paths and page-view durations) [Drott 1998; Fuller and de Graaff 1996; Sullivan 1997; Theng and Marsden 1998]. Some methods require manual pre-processing or filtering of the logs before analysis. Furthermore, the evaluator must interpret reported measures to identify usability problems. Software tools, such as WebTrends [WebTrends Corporation 2000], facilitate analysis by presenting results in graphical and report formats.

Statistical analysis is largely inconclusive for Web server logs, since they provide only a partial trace of user behavior and timing estimates may be skewed by network latencies. Server log files are also missing valuable information about what tasks users want to accomplish [Byrne *et al.* 1999; Schwartz 2000]. Nonetheless, statistical analysis techniques have been useful for improving usability and enable ongoing, cost-effective evaluation throughout the life of a site [Fuller and de Graaff 1996; Sullivan 1997].

Visualization is also used for inferential analysis of WIMP and Web log files [Chi *et al.* 2000; Cugini and Scholtz 1999; Guzdial *et al.* 1994; Hochheiser and Shneiderman 2001]. It enables evaluators to filter, manipulate, and render log file data in a way that ideally facilitates analysis. [Guzdial *et al.* 1994] propose several techniques for analyzing WIMP log files, such as color coding patterns and command usage, tracking screen updates, and tracking mouseclick locations and depth (i.e., number of times the user clicked the mouse in screen areas). However, there is no discussion of how effective these approaches are in supporting analysis.

Starfield visualization [Hochheiser and Shneiderman 2001] is one approach that enables evaluators to interactively explore Web server log data to gain an understanding of human factors issues related to visitation patterns. This approach combines the simultaneous display of a large number of individual data points (e.g., URLs requested versus time of requests) in an interface that supports zooming, filtering, and dynamic querying [Ahlberg and Shneiderman 1994]. Visualizations provide a high-level view of usage patterns (e.g., usage frequency, correlated references, bandwidth

usage, HTTP errors, and patterns of repeated visits over time) that the evaluator must explore to identify usability problems. It would be beneficial to employ a statistical analysis approach to study traffic, clickstreams, and page views prior to exploring visualizations.

The Dome Tree visualization [Chi *et al.* 2000] provides an insightful representation of simulated (see Section 2.9) and actual Web usage captured in server log files. This approach maps a Web site into a three dimensional surface representing the hyperlinks (see top part of Figure 2.5). The location of links on the surface is determined by a combination of content similarity, link usage, and link structure of Web pages. The visualization highlights the most commonly traversed subpaths. An evaluator can explore these usage paths to possibly gain insight about the information "scent" (i.e., common topics among Web pages on the path) as depicted in the bottom window of Figure 2.5. This additional information may help the evaluator infer what the information needs of site users are, and more importantly, may help infer whether the site satisfies those needs. The Dome Tree visualization also reports a crude path traversal time based on the sizes of pages (i.e., number of bytes in HTML and image files) along the path. Server log accuracy limits the extent to which this approach can successfully indicate usability problems. As is the case for Starfield visualization, it would be beneficial to statistically analyze log files prior to using this approach.

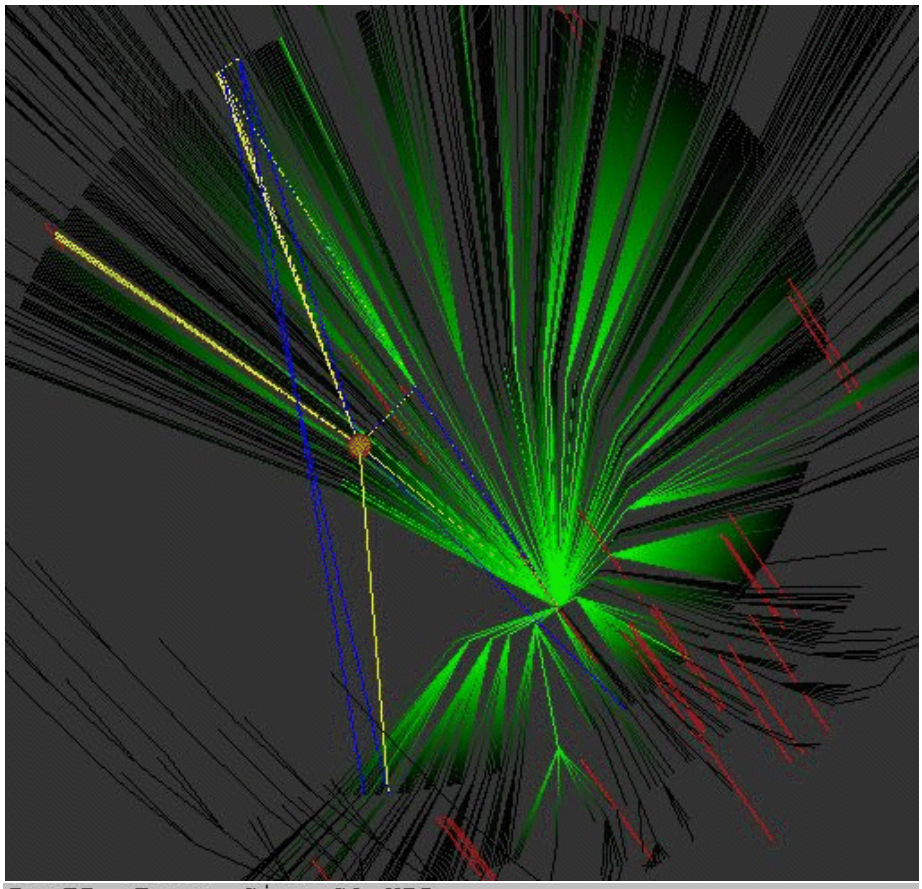
VISVIP [Cugini and Scholtz 1999] is a three-dimensional tool for visualizing log files compiled by WebVIP during usability testing (see previous section). Figure 2.6 shows VISVIP's Web site (top graph) and usage path (bottom graph) depictions to be similar to the Dome Tree visualization approach. VISVIP generates a 2D layout of the site where adjacent nodes are placed closer together than non-adjacent nodes. A third dimension reflects timing data as a dotted vertical bar at each node; the height is proportional to the amount of time. VISVIP also provides animation facilities for visualizing path traversal. Since WebVIP logs reflect actual task completion, prior statistical analysis is not necessary for VISVIP usage.

Usability Testing Methods: Automated Analysis – Discussion

Table 2.6 summarizes log file analysis methods discussed in this section. Although the techniques vary widely on the four assessment criteria (effectiveness, ease of use, ease of learning, and applicability), all approaches offer substantial benefits over the alternative – time-consuming, unaided analysis of potentially large amounts of raw data. Hybrid task-based pattern-matching techniques like USINE may be the most effective (i.e., provide clear insight for improving usability via task analysis), but they require additional effort and learning time over simpler pattern-matching approaches; this additional effort is mainly in the development of task models. Although pattern-matching approaches are easier to use and learn, they only detect problems for pre-specified usage patterns.

Metric-based approaches in the WIMP domain have been effective at associating measurements with specific interface aspects, such as commands and tasks, which can then be used to identify usability problems. AMME also helps the evaluator to understand the user's problem solving process and conduct simulation studies. However, metric-based approaches require the evaluator to conduct more analysis to ascertain the source of usability problems than task-based approaches. Metric-based techniques in the Web domain focus on server and network performance, which provides little usability insight. Similarly, inferential analysis of Web server logs is limited by their accuracy and may provide inconclusive usability information.

Most of the techniques surveyed in this section could be applied to WIMP and Web UIs other than those demonstrated on, with the exception of the MIKE UIMS and UsAGE, which require a WIMP UI to be developed within a special environment. AMME could be employed for both Web and WIMP UIs, provided log files and system models are available.



DocID	Freq	Size	SA	URL
33775	63	2445		docs/investor/sharinfo.htm
36369	168	3028		docs/factbook/1998/index.htm
33775	63	2445		docs/investor/sharinfo.htm
33778	58	20537		docs/investor/sitemap.htm
33780	26	2313		docs/investor/orderdoc.htm

Total Path Bytes: 30768 (17 Seconds)
Keywords for 33775 36369 33775 33778 33780:
reinvestment 7.230992
stock 6.651224
brochure 6.307321
dividend 5.522883
shareholder 5.165537
preferred 4.850075
automatic 4.215094
contacts 4.163801

Figure 2.5: Dome Tree visualization [Chi *et al.* 2000] of a Web site with a usage path displayed as a series of connected lines across the left side. The bottom part of the figure displays information about the usage path, including an estimated navigation time and information scent (i.e., common keywords along the path). Reprinted with permission of the authors.

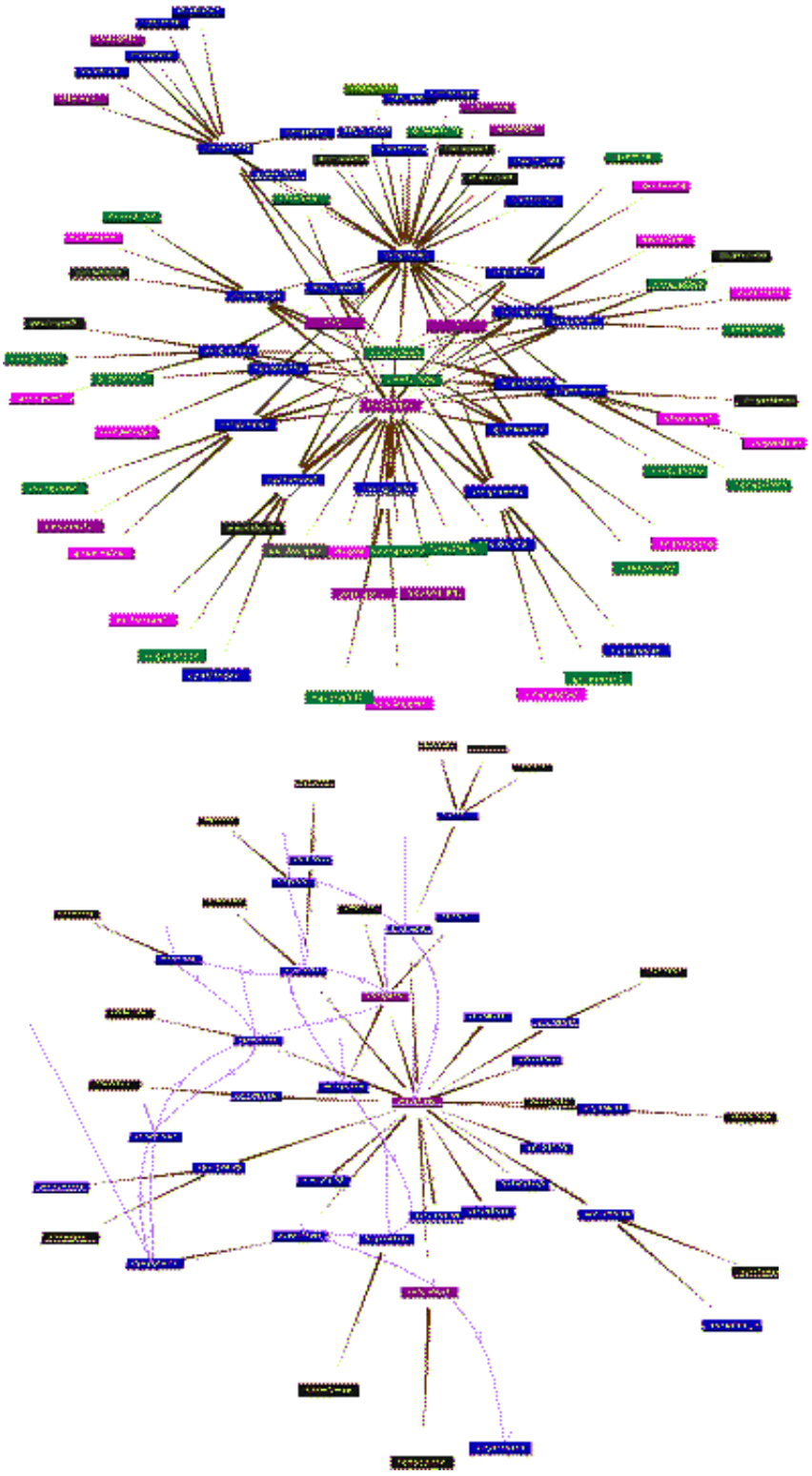


Figure 2.6: VISVIP visualization [Cugini and Scholtz 1999] of a Web site (top part). The bottom part of the figure displays a usage path (series of directed lines on the left site) laid over the site. Reprinted with permission of the authors.

2.6 Inspection Methods

A usability inspection is an evaluation methodology whereby an evaluator examines the usability aspects of a UI design with respect to its conformance to a set of guidelines. Guidelines can range from highly specific prescriptions to broad principles. Unlike other UE methods, inspections rely solely on the evaluator's judgment. A large number of detailed usability guidelines have been developed for WIMP interfaces [Open Software Foundation 1991; Smith and Mosier 1986] and Web interfaces [Comber 1995; Detweiler and Omanson 1996; Levine 1996; Lynch and Horton 1999; Web Accessibility Initiative 1999]. Common non-automated inspection techniques are heuristic evaluation [Nielsen 1993] and cognitive walkthroughs [Lewis *et al.* 1990].

Designers have historically experienced difficulties following design guidelines [Borges *et al.* 1996; de Souza and Bevan 1990; Lowgren and Nordqvist 1992; Smith 1986]. One study has demonstrated that designers are biased towards aesthetically pleasing interfaces, regardless of efficiency [Sears 1995]. Because designers have difficulty applying design guidelines, automation has been predominately used within the inspection class to objectively check guideline conformance. Software tools assist evaluators with guideline review by automatically detecting and reporting usability violations and in some cases making suggestions for fixing them [Balbo 1995; Farenc and Palanque 1999]. Automated capture, analysis, and critique support is available for the guideline review and cognitive walkthrough method types, as described in the remainder of this section.

2.6.1 Inspection Methods: Non-automated

Nine inspection method types were surveyed; they vary based on the goals of the inspection (e.g., guideline and standard conformance), the evaluative criteria (e.g., guidelines and standards), the evaluative process (e.g., formal or informal, task-based, or self-guided exploration), and how judgment is derived (e.g., individually or as a group). The fundamental goal of all inspection methods is to find usability problems in an existing interface design and then use these problems to make recommendations for improving the usability of an interface. Each inspection method has more specific objectives that aide in choosing the most appropriate method. For example, if the goal of a usability evaluation is to determine an interface's conformance to established guidelines, then the evaluator would use the guideline review method.

The nine inspection method types also differ in usability from the evaluator's perspective (i.e., how easy it is to learn and apply a method). Heuristic and perspective-based evaluations are considered to be easy to learn and apply, while cognitive walkthrough is not as easy to learn or apply [Nielsen and Mack 1994; Zhang *et al.* 1998]. Studies have also shown that the simpler the technique, the more effective the method is for identifying usability problems [Nielsen and Mack 1994]. For example, several studies have contrasted heuristic evaluation and cognitive walkthrough and reported heuristic evaluation to be more effective [Nielsen and Mack 1994].

In most cases, the method type is the same as the method. All of the methods require formal or informal interface usage and have been applied or could be applied to both WIMP and Web UIs. Unless otherwise specified, most discussions below are based on [Dix *et al.* 1998; Hom 1998; Human Factors Engineering 1999b; Nielsen 1993; Nielsen and Mack 1994; Shneiderman 1998].

Guideline Review. In guideline reviews evaluators check a WIMP interface for conformance with a comprehensive and sometimes large number (e.g., 1000 or more) of established usability guidelines. There are several accepted guidelines, including the Smith and Mosier guidelines [Smith and Mosier 1986] and the Motif style guides [Open Software Foundation 1991].

Guideline review is also actively used to evaluate Web interfaces. Many corporations have developed their own guidelines and there have been several efforts to develop a standard set of guidelines. Keevil [1998] presents a set of guidelines as a list of yes/no questions about the 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. Ambühler and Lindenmeyer [1999] use guidelines to compute accessibility measurements (i.e., how easy is it to access a page without special hardware or software). Lohse and Spiller [1998] 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.* [1999] propose guidelines to assist designers with determining the best navigation structure for a site. Currently, all of these approaches require manual evaluation.

Ratner *et al.* [1996] question the validity of HTML usability guidelines, since most HTML guidelines have not been subjected to a rigorous development process as established guidelines for WIMP interfaces. 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 WIMP guidelines existed in the 21 HTML style guides.

Cognitive Walkthrough. Cognitive walkthrough involves one or more evaluators exploring an interface, prototype, or paper mock-up by going through a pre-determined set of tasks and assessing the understandability and ease of learning for each task. During the walkthrough of a task, the evaluator(s) attempts to simulate a user's problem-solving process while examining each action required. The evaluator attempts to construct a credible success story for each step of the task. Otherwise, the evaluator constructs a detailed failure story.

Cognitive walkthroughs require intensive documentation effort. A modified version, cognitive jogthrough [Rowley and Rhoades 1992], was developed to expedite recording the walkthrough session. In cognitive jogthrough, the session is videotaped and logging software is used to mark key events. Thus, the videotape can be reviewed afterwards to document the session.

Pluralistic Walkthrough. This is a variation of the cognitive walkthrough inspection method wherein representative users, evaluators, and developers inspect the interface as a group. The goal of this method is to step through usage scenarios and discuss usability issues that arise in the scenario steps.

Heuristic Evaluation. In heuristic evaluation one or more evaluators independently evaluate an interface using a list of heuristics. The outcome of this evaluation is typically a list of possible usability problems. After the evaluators independently evaluate the interface, the evaluators aggregate their findings and associate severity ratings with each potential usability problem. Heuristic evaluation is the most informal inspection method [Nielsen and Mack 1994], mainly because it relies on a small set of usability criteria. It is also one of the main discount (i.e., cheap, fast, and easy to use) usability methods employed [Nielsen and Mack 1994].

Perspective-based Inspection. Perspective-based inspection [Zhang *et al.* 1998] is a variation of heuristic evaluation. For this method, evaluators divide a list of usability issues into different perspectives and focus on only one perspective or subset of heuristics during an inspection session. A perspective is a point of view consisting of a list of inspection questions and a specific procedure for conducting the inspection. Zhang *et al.* [1998] have shown that this

Method Class:	Inspection	
Automation Type:	Capture	
Method Type:	Cognitive Walkthrough - expert simulates user's problem solving (1 method)	
UE Method	**UI**	**Effort**
Software assists the expert with documenting a cognitive walkthrough	WIMP	F

Table 2.7: Synopsis of automated capture support for inspection methods.

approach improves the effectiveness of evaluators within each perspective as well as overall, in comparison to heuristic evaluation.

Feature Inspection. The purpose of this evaluation method is to inspect a feature set of a product and to analyze the availability, understandability, and other functionality aspects for each feature. Evaluators use a list of product features along with scenarios for such inspections. The documentation staff usually conducts feature inspections.

Formal Usability Inspection. Formal usability inspection is an adaptation of traditional software inspection to usability evaluation. The inspection procedure is fairly similar to heuristic evaluation and involves a diverse team of inspectors (e.g., developers, designers, documenters, trainers, technical support personnel, and possibly usability experts). The only difference is the formality associated with the inspection (i.e., assigned roles and a formal six-step process to follow).

Consistency Inspection. Evaluators use this method to determine a consistent interface appearance and functionality that they can then use to assess the consistency of interfaces across multiple products in a family.

Standards Inspection. In this inspection method an evaluator compares components of an interface to a list of industry standards to assess the interface's compliance with these standards. This inspection method is usually aimed at ensuring a product's market conformance.

2.6.2 Inspection Methods: Automated Capture

Table 2.7 summarizes capture support for inspection methods – namely, a system developed to assist an evaluator with a cognitive walkthrough. During a cognitive walkthrough, an evaluator attempts to simulate a user's problem-solving process while examining UI tasks. At each step of a task, the evaluator assesses whether a user would succeed or fail to complete the step. Hence, the evaluator produces extensive documentation during this analysis. There was an early attempt to "automate" cognitive walkthroughs by prompting evaluators with walkthrough questions and enabling evaluators to record their analyses in HyperCard. Unfortunately, evaluators found this approach too cumbersome and time-consuming to employ [Rieman *et al.* 1991].

2.6.3 Inspection Methods: Automated Analysis

Table 2.8 provides a synopsis of automated analysis methods for inspection-based usability evaluation, discussed in detail in the remainder of this section. All of the methods require minimal effort to employ; this is denoted with a blank entry in the effort column. Support available for WIMP and Web UIs is discussed separately.

Method Class:	Inspection	
Automation Type:	Analysis	
Method Type:	Guideline Review - expert checks guideline conformance (8 methods)	
UE Method	**UI**	**Effort**
Use quantitative screen measures for analysis (AIDE, [Parush *et al.* 1998])	WIMP	
Analyze terminology and consistency of UI elements (Sherlock)	WIMP	
Analyze the structure of Web pages (Rating Game, HyperAT, Gentler)	Web	
Use guidelines for analysis (WebSAT)	Web	
Analyze the scanning path of a Web page (Design Advisor)	Web	

Table 2.8: Synopsis of automated analysis support for inspection methods.

Inspection Methods: Automated Analysis – WIMP UIs

Several quantitative measures have been proposed for evaluating 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," "hot-spot," 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 [1996a] 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; Kim and Foley 1993] as well as several automated analysis tools [Mahajan and Shneiderman 1997; Parush *et al.* 1998; Sears 1995], discussed immediately below.

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.

Inspection Methods: Automated Analysis – Web UIs

The Rating Game [Stein 1997] is an automated analysis tool that attempts to measure the quality of a set of Web pages using a set of easily measurable features. These 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 improving a 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 the Hypertext Authoring Tool (HyperAT) is to support the creation of well-structured hyperdocuments. It provides a structural analysis which focuses on verifying that the breadths and depths within a page and at the site level fall within thresholds (e.g., depth less than three levels). (HyperAT also supports inferential analysis of server log files similarly to other log file analysis techniques; see Section 2.5.3.) Gentler [Thimbleby 1997] provides similar structural analysis but focuses on maintenance of existing sites rather than design of new ones.

The Web Static Analyzer Tool (SAT) [Scholtz and Laskowski 1998], part of the NIST WebMetrics suite of tools, assesses static HTML according to a number of usability guidelines, such as whether all graphics contain ALT tags, the average number of words in link text, and the existence of at least one outgoing link on a page. Currently, WebSAT only processes individual pages and does not suggest improvements [Chak 2000]. Future plans for this tool include adding the ability to inspect the entire site more holistically to identify potential problems in interactions between pages.

Unlike other analysis approaches, the Design Advisor [Faraday 2000] enables visual analysis of Web pages. The tool 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. The Design Advisor determines and superimposes a scanning path on a Web page where page elements are numbered to indicate the order in which elements will be scanned. It currently does not provide suggestions for improving scanning paths.

Inspection Methods: Automated Analysis – Discussion

Table 2.8 summarizes automated analysis methods discussed in this section. All of the 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. For example, [Farenc *et al.* 1999] 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.

The Rating Game, HyperAT, and Gentler compute and report a number of statistics about a page (e.g., number of links, graphics, and words). However, the effectiveness of these structural analyses is questionable, since the thresholds have not been empirically validated. Although there

Method Class:	Inspection	
Automation Type:	Critique	
Method Type:	Guideline Review - expert checks guideline conformance (11 methods)	
UE Method	**UI**	**Effort**
Use guidelines for critiquing (KRI/AG, IDA, CHIMES, Ergoval)	WIMP	
Use guidelines for critiquing and modifying a UI (SYNOP)	WIMP	M
Check HTML syntax (Weblint, Dr. Watson)	Web	
Use guidelines for critiquing (Lift Online, Lift Onsite, Bobby, WebEval)	Web	

Table 2.9: Synopsis of automated critique support for inspection methods.

have been some investigations into breadth and depth tradeoffs for the Web [Larson and Czerwinski 1998; Zaphiris and Mtei 1997], general thresholds still remain to be established. Although WebSAT helps designers adhere to good coding practices, these practices have not been shown to improve usability. There may be some indirect support for these methods through research aimed at identifying aspects that affect Web site credibility [Fogg *et al.* 2001; Kim and Fogg 1999; Fogg *et al.* 2000], since credibility affects usability and vice versa. A survey of over 1,400 Web users as well as an empirical study indicated that typographical errors, ads, broken links, and other aspects impact credibility; some of these aspects can be detected by automated UE tools, such as WebSAT. All of these approaches are easy to use, learn, and apply to all Web UIs.

The visual analysis supported by the Design Advisor could help designers improve Web page scanning. It requires a special Web browser for use, but is easy to use, learn, and apply to basic Web pages (i.e., pages that don't use scripts, applets, Macromedia Flash, or other non-HTML technology). Heuristics employed by this tool were developed based on empirical results from eye tracking studies of multimedia presentations, but have not been empirically validated for Web pages.

2.6.4 Inspection Methods: Automated Critique

Critique systems give designers clear directions for conforming to violated guidelines and consequently improving usability. As mentioned above, following guidelines is difficult, especially when there is a large number of guidelines to consider. Automated critique approaches, especially ones that modify a UI [Balbo 1995], provide the highest level of support for adhering to guidelines.

Table 2.9 provides a synopsis of automated critique methods discussed in the remainder of this section. All but one method, SYNOP, require minimal effort to employ; this is denoted with a blank entry in the effort column. Support available for WIMP and Web UIs is discussed separately.

Inspection Methods: Automated Critique – WIMP UIs

The KRI/AG tool (Knowledge-based Review of user Interface) [Lowgren and Nordqvist 1992] is an automated critique system that checks the guideline conformance of X Window interface designs created using the TeleUSE UIMS [Lee 1997]. KRI/AG 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 flaws in the design. IDA (user

Interface Design Assistance) [Reiterer 1994] also embeds rule-based (i.e., expert system) guideline checks within a UIMS. SYNOP [Balbo 1995] is a similar automated critique 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.

Inspection Methods: Automated Critique – Web UIs

Several automated critique tools use guidelines for Web site usability checks. The World Wide Web Consortium's HTML Validation Service [World Wide Web Consortium 2000] checks that HTML code conforms to standards. Weblint [Bowers 1996] and Dr. Watson [Addy & Associates 2000] also check HTML syntax and in addition verify links. Dr. Watson also computes download speed and spell checks text.

UsableNet's LIFT Online and LIFT Onsite [Usable Net 2000] perform usability checks similarly to WebSAT (discussed in Section 2.6.3) as well as checking for use of standard and portable link, text, and background colors, the existence of stretched images, and other guideline violations. LIFT Online suggests improvements, while LIFT Onsite guides users through making suggested improvements. According to [Chak 2000], these two tools provide valuable guidance for improving Web sites. Bobby [Clark and Dardailler 1999; Cooper 1999] is another HTML analysis tool that checks Web pages for their accessibility [Web Accessibility Initiative 1999] to people with disabilities.

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. As previously discussed, research on Web site credibility [Fogg *et al.* 2001; Kim and Fogg 1999; Fogg *et al.* 2000] 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 WIMP interfaces. 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 WIMP guidelines existed in the 21 HTML style guides. WebEval [Scapin *et al.* 2000] is one automated critique approach being developed to address this issue. Similarly to Ergoval (discussed above), it provides a framework for applying established WIMP guidelines to relevant HTML components. Even with WebEval, some problems, such as whether text will be understood by users, are difficult to detect automatically.

Inspection Methods: Automated Critique – Discussion

Table 2.9 summarizes automated critique methods discussed in this section. All of the WIMP approaches are highly effective at suggesting UI improvements for those guidelines that can be operationalized. These include checking for the existence of labels for text fields, listing menu

options in alphabetical order, and setting default values for input fields. However, they cannot assess UI aspects that cannot be operationalized, such as whether the labels used on elements will be understood by users. As previously discussed, [Farenc *et al.* 1999] 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. 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.

As previously discussed, conforming to the guidelines embedded in HTML analysis tools can potentially eliminate usability problems that arise due to poor HTML syntax (e.g., missing page elements) or guideline violations. 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 WIMP interfaces and have little consistency among them. Brajnik [2000] surveyed eleven automated Web site analysis methods, including Bobby and Lift Online. The author's survey revealed that these tools address only a sparse set of usability features, such as download time, presence of alternative text for images, and validation of HTML and links. Other usability aspects, such as consistency and information organization are unaddressed by existing tools.

All of the Web critique tools are applicable to basic HTML pages and appear to be easy to use and learn. They also enable ongoing assessment, which can be extremely beneficial after making changes.

2.7 Inquiry Methods

Similarly to usability testing approaches, inquiry methods require feedback from users and are often employed during usability testing. However, the focus is not on studying specific tasks or measuring performance. Rather the goal of these methods is to gather subjective impressions (i.e., preferences or opinions) about various aspects of a UI. Evaluators also use inquiry methods, such as surveys, questionnaires, and interviews, to gather supplementary data after a system is released; this is useful for improving the interface for future releases. In addition, evaluators use inquiry methods for needs assessment early in the design process.

Inquiry methods vary based on whether the evaluator interacts with a user or a group of users or whether users report their experiences using questionnaires or usage logs, possibly in conjunction with screen snapshots. Automation has been used predominately to capture subjective impressions during formal or informal interface use.

2.7.1 Inquiry Methods: Non-automated

This section provides a synopsis of non-automated inquiry method types. The method type and method are the same in all cases. All of the methods require formal or informal interface use. In addition, all of the methods have been or could be applied to WIMP and Web UIs. Unless otherwise specified, most discussions are based on [Dix *et al.* 1998; Hom 1998; Human Factors Engineering 1999b; Nielsen 1993; Shneiderman 1998].

Contextual Inquiry. Contextual inquiry is a structured field interviewing method based on three core principles: 1. understanding the context in which a product is used is essential for its successful design; 2. the user is a partner in the design process; and 3. the usability design process must have a focus. Given these guiding principles, an evaluator attempts to discover

users' needs through on-site free-flow interviewing. A contextual inquiry is usually a long-term study possibly lasting a year or more. It is usually conducted in the early stages of system development.

Field Observation. Field observation is similar to, but less structured than contextual inquiry. Furthermore, a field observation is typically conducted for a released system. For this method, evaluators visit the representative users' workplace and observe them working with the system. This enables the evaluator to understand how users are using the system to accomplish their tasks as well as the mental model the users have of the system. During this visit, the evaluator may also interview users about their jobs and other aspects about the way they use the product. Furthermore, evaluators may collect artifacts. This method is also described as an ethnographic study.

Focus Groups. A focus group is a meeting of about six to nine users wherein users discuss issues relating to the system. The evaluator plays the role of the moderator (i.e., asks about pre-determined issues) and gathers the needed information from the discussion. This is valuable for improving the usability of future releases.

Interviews. An interview is essentially a discussion session between a single user and an interviewer. During an interview, an evaluator asks a user a series of questions about system issues to guide the discussion. The evaluator can use either an unstructured or structured interviewing method. In unstructured interviewing there is no well-defined agenda, and the objective is to obtain information on procedures adopted by the user and the user's expectations of the system. Structured interviewing has a specific, pre-determined agenda with specific questions to guide and direct the interview. Unstructured interviewing is more of a conversation, while structured interviewing is more of an interrogation.

Surveys. During a survey, an evaluator asks a user pre-determined questions and records responses. However, surveys are not as formal or structured as interviews.

Questionnaires. A questionnaire is a measurement tool designed to assess a user's subjective satisfaction with an interface. It is a list of questions that are distributed to users for responses. Responses on a questionnaire are usually quantitative (e.g., ratings on a 5-point scale). One example questionnaire is the Questionnaire for User Interaction Satisfaction (QUIS) [Harper and Norman 1993; Human Factors Engineering 1999b]. QUIS contains questions to rate 27 system attributes on a 10-point scale, including overall system satisfaction, screen visibility, terminology, system information, learning factors, and system capabilities.

Self-reporting Logs. Self-reporting logs is a paper-and-pencil form of logging wherein users write down their actions, observations, and comments on a system and then send them to the evaluator. This method is most appropriate in early stages of development.

Screen Snapshots. Screen snapshots are usually captured by a participant in conjunction with other journaling methods, such as self-reporting logs. Basically, participants take screen snapshots at various times during execution of pre-determined tasks.

User Feedback. User feedback is a means for users to give comments on the system as necessary or at their convenience. For some systems, it may be possible to make a feedback button or command accessible within the interface. It is also possible to allow users to submit feedback via electronic mail, bulletin boards, or Web sites.

Method Class:	Inquiry	
Automation Type:	Capture	
Method Type:	Questionnaires - user provides answers to specific questions (2 methods)	
UE Method	**UI**	**Effort**
Questionnaire embedded within the UI (UPM)	WIMP	IF
HTML forms-based questionnaires (e.g., WAMMI, QUIS, SUMI, or NetRaker)	WIMP, Web	IF

Table 2.10: Synopsis of automated capture support for inquiry methods.

2.7.2 Inquiry Methods: Automated Capture

Table 2.10 provides a synopsis of capture methods developed to assist users with completing questionnaires. Software tools enable the evaluator to collect subjective usability data and possibly make improvements throughout the life of an interface. Questionnaires can be embedded within a WIMP UI to facilitate the response capture process. Typically dialog boxes prompt users for subjective input and process responses (e.g., save data to a file or email data to the evaluator). For example, UPM (the User Partnering Module) [Abelow 1993] uses event-driven triggers (e.g., errors or specific command invocations) to ask users specific questions about their interface usage. This approach allows the evaluator to capture user reactions while they are still fresh.

The Web inherently facilitates the capture of questionnaire data using forms. Users are typically presented with an HTML page for entering data, and a program on the Web server (e.g., a CGI script) processes responses. Several validated questionnaires are available in Web format, including QUIS (Questionnaire for User Interaction Satisfaction) [Harper and Norman 1993] and SUMI (Software Usability Measurement Inventory) [Porteous *et al.* 1993] for WIMP interfaces and WAMMI (Website Analysis and MeasureMent Inventory) [Kirakowski and Claridge 1998] for Web interfaces. NetRaker's [NetRaker 2000] usability research tools enable evaluators to create custom HTML questionnaires and usability tests via a template interface and to view a graphical summary of results even while studies are in progress. NetRaker's tools include the NetRaker Index (a short usability questionnaire) for continuously gathering feedback from users about a Web site. Chak [2000] reports that NetRaker's tools are highly effective for gathering direct user feedback, but points out the need to address potential irritations caused by the NetRaker Index's pop-up survey window.

As previously discussed, automated capture methods represent an important first step toward informing UI improvements. Automation support for inquiry methods makes it possible to collect data quickly from a larger number of users than is typically possible without automation. However, these methods suffer from the same limitation of non-automated approaches – they may not clearly indicate usability problems due to the subjective nature of user responses. Furthermore, they do not support automated analysis or critique of interfaces. The real value of these techniques is that they are easy to use and widely applicable.

2.8 Analytical Modeling Methods

Analytical modeling complements traditional evaluation techniques like usability testing. Given some representation or model of the UI and/or the user, these methods enable the evaluator to inexpensively predict usability. A wide range of modeling techniques have been developed, and

they support different types of analyses. de Haan *et al.* [1992] classify modeling approaches into the following four categories:

- **Models for task environment analysis:** enable the evaluator to assess the mapping between the user's goals and UI tasks (i.e., how the user accomplishes these goals within the UI). ETIT (External Internal Task Mapping) [Moran 1983] is one example for evaluating the functionality, learnability, and consistency of the UI;
- **Models to analyze user knowledge:** enable the evaluator to use formal grammars to represent and assess knowledge required for interface use. AL (Action Language) [Reisner 1984] and TAG (Task-Action Grammar) [Payne and Green 1986] allow the evaluator to compare alternative designs and predict differences in learnability;
- **Models of user performance:** enable the evaluator to predict user behavior, mainly task completion time. GOMS analysis (Goals, Operators, Methods, and Selection rules) [John and Kieras 1996], CTA (Cognitive Task Analysis) [May and Barnard 1994], and (Programmmable User Models) PUM [Young *et al.* 1989] – support performance prediction; and
- **Models of the user interface:** enable the evaluator to represent the UI design at multiple levels of abstraction (e.g., syntactic and semantic levels) and assess this representation. CLG (Command Language Grammar) [Moran 1981] and ETAG (Extended Task-Action Grammar) [Tauber 1990] are two methods for representing and inspecting designs.

Models that focus on user performance, such as GOMS analysis, typically support quantitative analysis. The other approaches typically entail qualitative analysis and in some cases, such as TAG, support quantitative analysis as well. The survey only revealed automation support for methods that focus on user performance, including GOMS analysis, CTA, and PUM; this is most likely because performance prediction methods support quantitative analysis, which is easier to automate.

Automation has been predominately used to analyze task completion (e.g., execution and learning time) within WIMP UIs. Analytical modeling inherently supports automated analysis. The survey did not reveal analytical modeling techniques to support automated critique. Most analytical modeling and simulation approaches are based on the model human processor (MHP) proposed by Card *et al.* [1983]. GOMS analysis (Goals, Operators, Methods, and Selection Rules) is one of the most widely accepted analytical modeling methods based on the MHP [John and Kieras 1996]. Other methods based on the MHP employ simulation and will be discussed in Section 2.9.

2.8.1 Analytical Modeling Methods: Non-automated

This section provides a synopsis of non-automated modeling method types. Method types and methods are the same in all cases. All of the methods require model development and have only been employed for WIMP interfaces.

GOMS Analysis. The GOMS family of analytical modeling 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 enable predictions of task execution and learning times, typically for error-free expert performance. The four approaches in this family include the original GOMS method proposed by Card, Moran, and Newell (CMN-GOMS) [Card *et al.* 1983], the simpler keystroke-level model (KLM), the natural GOMS language (NGOMSL), and the critical path method (CPM-GOMS) [John and Kieras 1996].

These approaches differ in the task granularity modeled (e.g., keystrokes versus a high-level procedure), the support for alternative task completion methods, and the support for single goals versus multiple simultaneous goals.

Task-Environment Analysis. ETIT (External Internal Task Mapping) [Moran 1983] is an example method for studying the relationship between tasks completed in the user's domain and the mapping of these tasks into UI tasks. Terminology used for specific objects (e.g., character, word, or sentence) and operators on these objects (e.g., copy, split, or join) are first enumerated for each domain; then, mappings between the domains are determined. Establishing such a mapping enables the evaluator to make inferences about the functionality, learnability, and consistency of the UI. In addition, this method can be used for assessing the degree of knowledge transfer between alternative designs.

Knowledge Analysis. Knowledge analysis methods, such as AL (Action Language) [Reisner 1984] and TAG (Task-Action Grammar) [Payne and Green 1986], provide a formal grammar for representing and assessing knowledge required for converting specific user tasks into UI tasks. Both of these techniques assess usability by counting the number and depth of rules, but differ with respect to the formal grammar employed. AL uses a well-known notation for expressing computer science programming languages – Backus-Naur Form [Backus *et al.* 1964]. TAG uses a more sophisticated grammar, which produces more compact representations of rules. Measures computed from AL and TAG task representations can be used to compare alternative designs and to predict differences in learnability.

Design Analysis. Design analysis methods, such as CLG (Command Language Grammar) [Moran 1981] and ETAG (Extended Task-Action Grammar) [Tauber 1990], enable the evaluator to represent the UI design at multiple levels of abstraction (e.g., syntactic and semantic levels) and assess this representation. These methods are typically used for design specification prior to UI implementation. ETAG is a refinement of CLG that supports additional levels of analysis, such as specifying the syntax. ETAG has also been used for modeling user performance and knowledge.

2.8.2 Analytical Modeling Methods: Automated Analysis

Table 2.11 provides a synopsis of automated analysis methods discussed in the remainder of this section. The survey did not reveal analytical modeling methods for evaluating Web UIs.

Analytical Modeling Methods: Automated Analysis – WIMP UIs

Two of the major roadblocks to using GOMS have been the tedious task analysis and the need to calculate execution and learning times [Baumeister *et al.* 2000; Byrne *et al.* 1994; Hudson *et al.* 1999; Kieras *et al.* 1995]. These were originally specified and calculated manually or with generic tools such as spreadsheets. In some cases, evaluators implemented GOMS models in computational cognitive architectures, such as Soar or EPIC (discussed in Section 2.9). This approach actually added complexity and time to the analysis [Baumeister *et al.* 2000]. QGOMS (Quick and dirty GOMS) [Beard *et al.* 1996] and CATHCI (Cognitive Analysis Tool for Human Computer Interfaces) [Williams 1993] provide support for generating quantitative predictions, but still require the evaluator to construct GOMS models. Baumeister *et al.* [2000] studied these approaches and showed them to be inadequate for GOMS analysis.

Method Class:	Analytical Modeling	
Automation Type:	Analysis	
Method Type:	UIDE Analysis - conduct GOMS analysis within a UIDE (4 methods)	
UE Method	**UI**	**Effort**
Generate predictions for GOMS task models (QGOMS, CATHCI)	WIMP	M
Generate GOMS task models and predictions (USAGE, CRITIQUE)	WIMP	M
Method Type:	Cognitive Task Analysis - predict usability problems (1 method)	
UE Method	**UI**	**Effort**
Cognitive Task Analysis (CTA)	WIMP	M
Method Type:	Programmable User Models - write program that acts like a user (1 method)	
UE Method	**UI**	**Effort**
Programmable User Models (PUM)	WIMP	M

Table 2.11: Synopsis of automated analysis support for analytical modeling methods.

USAGE[5] (the UIDE System for semi-Automated GOMS Evaluation) [Byrne *et al.* 1994] and CRITIQUE (the Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations) [Hudson *et al.* 1999] provide support for automatically generating a GOMS 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) [Kieras *et al.* 1995] is another tool that generates quantitative predictions for a given GOMS task model (discussed in more detail in Section 2.9). These tools reduce the effort required to employ GOMS analysis and generate predictions that are consistent with models produced by experts. The major hindrance to wide application of these tools is that they operate on limited platforms (e.g., Sun machines), model low-level goals (e.g., at the keystroke level for CRITIQUE), do not support multiple task completion methods (even though GOMS was designed to support this), and rely on an idealized expert user model.

Cognitive Task Analysis (CTA) [May and Barnard 1994] uses a different modeling approach than GOMS analysis. GOMS analysis requires the evaluator to construct a model for each task to be analyzed. However, CTA requires the evaluator to input an interface description to an underlying theoretical model for analysis. The theoretical model, an expert system based on Interacting Cognitive Subsystems (ICS, discussed in Section 2.9), generates predictions about performance and usability problems similarly to a cognitive walkthrough. The system prompts the evaluator for interface details from which it generates predictions and a report detailing the theoretical basis of predictions. The authors refer to this form of analysis as "supportive evaluation."

The Programmable User Model (PUM) [Young *et al.* 1989] is an entirely different analytical modeling technique. In this approach, the designer is required to write a program that acts like a user using the interface; the designer must specify explicit sequences of operations for each task. Task sequences are then analyzed by an architecture (similar to the CTA expert system) that imposes approximations of psychological constraints, such as memory limitations. Constraint violations can be seen as potential usability problems. The designer can alter the interface design to resolve violations, and ideally improve the implemented UI as well. Once the designer successfully programs the architecture (i.e., creates a design that adheres to the psychological constraints),

[5]This is not to be confused with the UsAGE log file capture and analysis tool discussed in Section 2.5.

the model can then be used to generate quantitative performance predictions similarly to GOMS analysis. By making a designer aware of considerations and constraints affecting usability from the user's perspective, this approach provides clear insight into specific problems with a UI.

Analytical Modeling Methods: Automated Analysis – Discussion

Table 2.11 summarizes automated analysis methods discussed in this section. Analytical modeling approaches enable the evaluator to produce relatively inexpensive results to inform design choices. GOMS analysis has been shown to be applicable to all types of WIMP UIs and is effective at predicting usability problems. However, these predictions are limited to error-free expert performance in many cases although early accounts of GOMS considered error correction [Card *et al.* 1983]. The development of USAGE and CRITIQUE has reduced the learning time and effort required to apply GOMS analysis, but they suffer from limitations previously discussed. Tools based on GOMS may also require empirical studies to determine operator parameters in cases where these parameters have not been previously validated and documented.

Although CTA is an ideal solution for iterative design, it does not appear to be a fully-developed methodology. Two demonstration systems have been developed and effectively used by a group of practitioners as well as by a group of graduate students [May and Barnard 1994]. However, some users experienced difficulty with entering system descriptions, which can be a time consuming process. After the initial interface specification, subsequent analysis is easier because the demonstration systems store interface information. The approach appears to be applicable to all WIMP UIs. It may be possible to apply a more fully developed approach to Web UIs.

PUM is a programming approach, and thus requires considerable effort and learning time to employ. Although it appears that this technique is applicable to all WIMP UIs, its effectiveness is not discussed in detail in the literature.

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). This addresses only a small fraction of usability problems. While analytical modeling techniques are potentially beneficial, the survey did not uncover any approaches that address this gap in Web site evaluation. Approaches like GOMS analysis will not map as well to the Web domain, because it is difficult to predict how a user will accomplish the goals in a task hierarchy given that there are potentially many different ways to navigate a typical site. Another problem is GOMS' reliance on an expert user model (at least in the automated approaches), which does not fit the diverse user community of the Web. Hence, new analytical modeling approaches, such as a variation of CTA, are required to evaluate the usability of Web sites.

2.9 Simulation Methods

Simulation complements traditional UE methods and inherently supports automated analysis. Using models of the user and/or the interface design, computer programs simulate the user interacting with the interface and report the results of this interaction, in the form of performance measures and interface operations, for instance. Evaluators can run simulators with different parameters to study various UI design tradeoffs and thus make more informed decisions about UI implementation. Simulation is also used to automatically generate synthetic usage data for analysis with log file analysis techniques [Chi *et al.* 2000] or event playback in a UI [Kasik and George 1996]. Thus, simulation can also be viewed as supporting automated capture to some degree.

Method Class:	Simulation	
Automation Type:	Capture	
Method Type:	Genetic Algorithm Modeling - mimic novice user interaction (1 method)	
UE Method	**UI**	**Effort**
Genetic Algorithm Modeling ([Kasik and George 1996])	WIMP	
Method Type:	Information Scent Modeling - mimic Web site navigation (1 method)	
UE Method	**UI**	**Effort**
Information Scent Modeling ([Chi *et al.* 2000])	Web	M

Table 2.12: Synopsis of automated capture support for simulation methods.

2.9.1 Simulation Methods: Automated Capture

Table 2.12 provides a synopsis of the two automated capture methods discussed in this section. Kasik and George [1996] developed an automated technique for generating and capturing usage data; this data could then be used for driving tools that replay events (such as executing a log file) within Motif-based UIs. The goal of this work is to use a small number of input parameters to inexpensively generate a large number of usage traces (or test scripts) representing novice users. The evaluator can then use these traces to find weak spots, failures, and other usability problems.

To create novice usage traces, the designer initially produces a trace representing an expert using the UI; a scripting language is available to produce this trace. The designer can then insert deviation commands at different points within the expert trace. During trace execution, a genetic algorithm determines user behavior at deviation points, and in effect simulates a novice user learning by experimentation. Genetic algorithms consider past history in generating future random numbers; this enables the emulation of user learning. Altering key features of the genetic algorithm enables the designer to simulate other user models. Although currently not supported by this tool, traditional random number generation can also be employed to explore the outer limits of a UI, for example, by simulating completely random behavior.

Chi *et al.* [2000] developed a similar approach for generating and capturing navigation paths for Web UIs. This approach creates a model of an existing site that embeds information about the similarity of content among pages, server log data, and linking structure. The evaluator specifies starting points in the site and information needs (i.e., target pages) as input to the simulator. The simulator models a number of agents (i.e., hypothetical users) traversing the links and content of the site model. At each page, the model considers information "scent" (i.e., common keywords between an agent's goal and content on linked pages) in making navigation decisions. Navigation decisions are controlled probabilistically such that most agents traverse higher-scent links (i.e., closest match to information goal) and some agents traverse lower-scent links. Simulated agents stop when they reach the target pages or after an arbitrary amount of effort (e.g., maximum number of links or browsing time). The simulator records navigation paths and reports the proportion of agents that reached target pages.

The authors use these usage paths as input to the Dome Tree visualization methodology, an inferential log file analysis approach discussed in Section 2.5. The authors compared actual and simulated navigation paths for Xerox's corporate site and discovered a close match when scent is "clearly visible" (meaning links are not embedded in long text passages or obstructed by images). Since the site model does not consider actual page elements, the simulator cannot account for the impact of various page aspects, such as the amount of text or reading complexity, on navigation

Method Class:	Simulation	
Automation Type:	Analysis	
Method Type:	Petri Net Modeling - mimic user interaction from usage data (1 method)	
UE Method	**UI**	**Effort**
Petri Net Modeling (AMME)	WIMP	IF
Method Type:	Information Processor Modeling - mimic user interaction (9 methods)	
UE Method	**UI**	**Effort**
Employ a computational cognitive architecture for UI analysis (ACT-R, COGNET, EPIC, HOS, Soar, CCT, ICS, GLEAN)	WIMP	M
Employ a GOMS-like model to analyze navigation (Site Profile)	Web	M

Table 2.13: Synopsis of automated analysis support for simulation methods.

choices. Hence, this approach may enable only crude approximations of user behavior for sites with complex pages.

Simulation Methods: Automated Capture – Discussion

Table 2.12 summarizes automated capture methods discussed in this section. Without these techniques, the evaluator must anticipate all possible usage scenarios or rely on formal or informal interface use to generate usage traces. Formal and informal use limit UI coverage to a small number of tasks or to UI features that are employed in regular use. Automated techniques, such as the genetic algorithm approach, enable the evaluator to produce a larger number of usage scenarios and widen UI coverage with minimal effort.

The system developed by Kasik and George appears to be relatively straightforward to use, since it interacts directly with a running application and does not require modeling. Interaction with the running application also ensures that generated usage traces are plausible. Experiments demonstrated that it is possible to generate a large number of usage traces within an hour. However, an evaluator must manually analyze the execution of each trace to identify problems. The authors propose future work to automatically verify that a trace produced the correct result. The evaluator must also program an expert user trace, which could make the system difficult to use and learn. Currently, this tool is only applicable to Motif-based UIs.

The approach developed by Chi *et al.* is applicable to all Web UIs. It also appears to be straightforward to use and learn, since software produces the Web site model automatically. The evaluator must manually interpret simulation results; however, analysis could be facilitated with the Dome Tree visualization tool.

2.9.2 Simulation Methods: Automated Analysis

Table 2.13 provides a synopsis of the automated analysis methods discussed in the remainder of this section. Methods for WIMP and Web UIs are considered separately.

Simulation Methods: Automated Analysis – WIMP UIs

AMME [Rauterberg and Aeppili 1995] (see Section 2.5.2) is the only surveyed approach that constructs a WIMP simulation model (petri net) directly from usage data. Other methods are based on a model similar to the MHP and require the evaluator to conduct a task analysis (and

subsequently validate it with empirical data) to develop a simulator. Hence, AMME is more accurate, flexible (i.e., task and user independent), and simulates more detail (e.g., error performance and preferred task sequences). AMME simulates learning, user decisions, and task completion and outputs a measure of behavior complexity. Studies have shown that the behavior complexity measure correlates negatively with learning and interface complexity. Studies have also validated the accuracy of generated models with usage data [Rauterberg 1995]. AMME should be applicable to Web interfaces as well, since it constructs models from log files. Despite its advantages, AMME still requires formal interface use to generate log files for simulation studies.

The remaining WIMP simulation methods are based on sophisticated computational cognitive architectures – theoretical models of user behavior – similar to the MHP previously discussed. Unlike analytical modeling approaches, these methods attempt to approximate user behavior as accurately as possible. For example, the simulator may track the user's memory contents, interface state, and the user's hand movements during execution. This enables the simulator to report a detailed trace of the simulation run. Some simulation methods, such as CCT [Kieras and Polson 1985] (discussed below), can also generate predictions statically (i.e., without being executed) similarly to analytical modeling methods.

Pew and Mavor [Pew and Mavor 1998] provide a detailed discussion of computational cognitive architectures and an overview of many approaches, including five discussed below: ACT-R (Adaptive Control of Thought) [Anderson 1990; Anderson 1993], COGNET (COGnition as a NEtwork of Tasks) [Zachary *et al.* 1996], EPIC (Executive-Process Interactive Control) [Kieras *et al.* 1997], HOS (Human Operator Simulator) [Glenn *et al.* 1992], and Soar [Laird and Rosenbloom 1996; Polk and Rosenbloom 1994]. Here, CCT (Cognitive Complexity Theory) [Kieras and Polson 1985], ICS (Interacting Cognitive Subsystems) [Barnard 1987; Barnard and Teasdale 1991], and GLEAN (GOMS Language Evaluation and ANalysis) [Kieras *et al.* 1995] are also considered. Rather than describe each method individually, Table 2.14 summarizes the major characteristics of these simulation methods as discussed below.

Modeled Tasks. The surveyed models simulate the following three 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 or 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 or system, simulation methods use either task hierarchies (as in a 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, ACT-R), memory load

Parameter	UE Methods
Modeled Tasks	
problem-solving and/or learning	COGNET, ACT-R, Soar, ICS
human-machine system	HOS
UI interaction	EPIC, GLEAN, CCT
Modeled Components	
cognition	ACT-R, COGNET
perception, cognition & motor	EPIC, ICS, HOS, Soar, GLEAN, CCT
Component Processing	
serial	ACT-R, GLEAN, CCT
semi-parallel	COGNET, HOS
parallel	EPIC, ICS, Soar
Model Representation	
task hierarchy	HOS, CCT
production rules	CCT, ACT-R, EPIC, Soar, ICS
program	GLEAN, COGNET
Predictions	
task performance	EPIC, CCT, COGNET, GLEAN, HOS, Soar, ACT-R
memory load	ICS, CCT
learning	ACT-R, Soar, ICS, GLEAN, CCT
behavior	ACT-R, COGNET, EPIC

Table 2.14: Characteristics of WIMP simulation methods that are based on a variation of the MHP.

(ICS, CCT), learning (ACT-R, SOAR, ICS, GLEAN, CCT), or behavior predictions such as action traces (ACT-R, COGNET, EPIC).

These methods vary widely in their ability to illustrate usability problems. Their effectiveness is largely determined by the characteristics discussed (modeled tasks, modeled components, component processing, model representation, and predictions). Methods that are potentially the most effective at illustrating usability problems model UI interaction and all components (perception, cognition, and motor) processing in parallel, employ production rules, and report on task performance, memory load, learning, and simulated user behavior. Such methods would enable the most flexibility and closest approximation of actual user behavior. The use of production rules is important in this methodology, because it relaxes the requirement for an explicit task hierarchy, thus allowing for the modeling of more dynamic behavior, such as Web site navigation.

EPIC is the only simulation analysis method that embodies most of these ideal characteristics. It uses production rules and models UI interaction and all components (perception, cognition, and motor) processing in parallel. It reports task performance and simulated user behavior, but does not report memory load and learning estimates. Studies with EPIC have demonstrated that predictions for telephone operator and menu searching tasks closely match observed data. EPIC and all of the other methods require considerable learning time and effort to use. They are also applicable to a wide range of WIMP UIs.

Simulation Methods: Automated Analysis – Web UIs

The survey revealed only one simulation approach for analysis of Web interfaces – WebCriteria's Site Profile [Web Criteria 1999]. Unlike the other simulation approaches, it requires an

implemented interface for evaluation. Site Profile performs analysis 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 nodes-and-links model of the site. For the analysis phase, it uses an idealistic Web user model (called Max [Lynch *et al.* 1999]) 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.

The usefulness of this approach is questionable, since currently it only computes accessibility (navigation time) for the shortest path between specified start and destination pages using a single user model. Other measurements, such as freshness and page composition, also have questionable value in improving the Web site. [Brajnik 2000] showed Site Profile to support only a small fraction of the analysis supported by guideline review methods, such as WebSAT and Bobby (discussed in Section 2.6). [Chak 2000] also cautions that the accessibility measure should be used as an initial benchmark, not a highly-accurate approximation. Site Profile does not entail any learning time or effort on the part of the evaluator, since WebCriteria performs the analysis. The method is applicable to all Web UIs.

Simulation Methods: Automated Analysis – Discussion

Table 2.13 summarizes automated analysis methods discussed in this section. Unlike most evaluation approaches, simulation can be used prior to UI implementation in most cases (although AMME and WebCriteria's Site Profile are exceptions to this). Hence, simulation enables alternative designs to be compared and optimized before implementation.

It is difficult to assess the effectiveness of simulation methods, although there have been reports that show EPIC [Kieras *et al.* 1997] and GLEAN [Baumeister *et al.* 2000] to be effective. AMME appears to be the most effective method, since it is based on actual usage. AMME also enables ongoing assessment and could be widely used for WIMP and Web interfaces, provided log files and system models are available. EPIC is the only method based on the MHP that embodies the ideal simulator characteristics previously discussed. GLEAN is actually based on EPIC, so it has similar properties.

In general, simulation methods are more difficult to use and learn than other evaluation methods, because they require constructing or manipulating complex models as well as understanding the theory behind a simulation approach. Approaches based on the MHP are widely applicable to all WIMP UIs. Approaches that use production rules, such as EPIC, CCT, and Soar, could possibly be applied to Web UIs where task sequences are not as clearly defined as WIMP UIs. Soar has actually been adapted to model browsing tasks similar to Web browsing [Peck and John 1992].

2.10 Expanding Existing Approaches to Automating Usability Evaluation Methods

Automated usability evaluation methods have many potential benefits, including reducing the costs of non-automated methods, aiding in comparisons between alternative designs, and improving consistency in evaluation results. Numerous methods that support automation have been studied. Based on the methods surveyed, research to further develop log file analysis, guideline review, analytical modeling, and simulation techniques could result in several promising automated

Method Class: Testing		
Automation Type: Analysis		
Method Type: Log File Analysis - analyze usage data (20 methods)		
UE Method	**UI**	**Effort**
Use metrics during log file analysis (DRUM, MIKE UIMS, AMME)	WIMP	IF
Use metrics during log file analysis (Service Metrics, [Bacheldor 1999])	Web	IF
Use pattern matching during log file analysis (MRP)	WIMP	IF
Use task models during log file analysis (IBOT, QUIP, WebQuilt, KALDI, UsAGE)	WIMP	IF
Use task models and pattern matching during log file analysis (ÉMA, USINE, RemUSINE)	WIMP	IFM
Visualization of log files ([Guzdial *et al.* 1994])	WIMP	IF
Statistical analysis or visualization of log files (traffic- and time-based analyses, VISVIP, Starfield and Dome Tree visualizations)	Web	IF

Table 2.15: Synopsis of automated analysis support for usability testing methods. This is a repetition of Table 2.6.

techniques as discussed in more detail below. Chapter 3 discusses other promising approaches based on performance evaluation of computer systems.

2.10.1 Expanding Log File Analysis Approaches

The survey showed log file analysis to be a viable methodology for automated analysis of usage data. Table 2.15 summarizes current approaches to log file analysis. These approaches could be expanded and improved in the following three ways:

- Generating synthetic usage data for analysis;
- Using log files for comparing (i.e., benchmarking) comparable UIs; and
- Augmenting task-based pattern-matching approaches with guidelines to support automated critique.

Generating synthetic usage data for analysis. The main limitation of log file analysis is that it still requires formal or informal interface use 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. This is even more important for Web interfaces, since server logs do not capture a complete record of user interactions. The discussion included two simulation approaches, one using a genetic algorithm [Kasik and George 1996] and the other using information scent modeling [Chi *et al.* 2000] (see Section 2.9.1), that automatically generate plausible usage data. The genetic algorithm approach determines user behavior during deviation points in an expert user script, while the information scent model selects navigation paths by considering word overlap between links and web pages. Both of these approaches generate plausible usage traces without formal or informal interface use. These techniques also provide valuable insight on how to leverage real usage data from usability tests or informal use. For example, real data could also serve as input scripts for genetic algorithms; the evaluator could add deviation points to these.

Method Class:	Inspection	
Automation Type:	Analysis	
Method Type:	Guideline Review - expert checks guideline conformance (8 methods)	
UE Method	**UI**	**Effort**
Use quantitative screen measures for analysis (AIDE, [Parush *et al.* 1998])	WIMP	
Analyze terminology and consistency of UI elements (Sherlock)	WIMP	
Analyze the structure of Web pages (Rating Game, HyperAT, Gentler)	Web	
Use guidelines for analysis (WebSAT)	Web	
Analyze the scanning path of a Web page (Design Advisor)	Web	

Table 2.16: Synopsis of automated analysis support for inspection methods. This is a repetition of Table 2.8.

Using log files for comparing UIs. Real and simulated usage data could also be used to evaluate comparable WIMP UIs, such as word processors and image editors. Task sequences could comprise a usability benchmark (i.e., a program for measuring UI performance); this is similar to GOMS analysis of comparable task models. After mapping task sequences into specific UI operations in each interface, the benchmark could be executed within each UI to collect measurements. Representing this benchmark as a log file of some form would enable the log file to be executed within a UI by replay tools, such as: QC/Replay [Centerline 1999] for X Windows; UsAGE [Uehling and Wolf 1995] for replaying events within a UIMS (discussed in Section 2.5); or WinRunner [Mercury Interactive 2000] for a wide range of applications (e.g., Java and Oracle applications). This is a promising open area of research for evaluating comparable WIMP UIs. Chapter 3 explores this concept in more detail.

Augmenting task-based pattern-matching approaches with guidelines to support automated critique. Given a wider sampling of usage data, using task models and pattern matching during 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. USINE outputs information to help the evaluator understand 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 to support automated critique.

2.10.2 Expanding Guideline Review Approaches

Several guideline review methods for analysis of WIMP interfaces (see Table 2.16) could be augmented with guidelines to support automated critique. For example, AIDE (discussed in Section 2.6) provides the most support for evaluating UI designs. It computes a number of quantitative measures and also generates initial interface layouts. Guidelines, such as thresholds for quantitative measures, could also be incorporated into AIDE analysis to support automated critique.

Although there are several guideline review methods for analyzing and critiquing Web UIs (see Tables 2.16 and 2.17), existing approaches only cover a small fraction of usability aspects [Brajnik 2000] and have not been empirically validated. This dissertation presents an approach for developing Web design guidelines directly from empirical data.

Method Class:	Inspection	
Automation Type:	Critique	
Method Type:	Guideline Review - expert checks guideline conformance (11 methods)	
UE Method	**UI**	**Effort**
Use guidelines for critiquing (KRI/AG, IDA, CHIMES, Ergoval)	WIMP	
Use guidelines for critiquing and modifying a UI (SYNOP)	WIMP	M
Check HTML syntax (Weblint, Dr. Watson)	Web	
Use guidelines for critiquing (Lift Online, Lift Onsite, Bobby, WebEval)	Web	

Table 2.17: Synopsis of automated critique support for inspection methods. This is a repetition of Table 2.9.

2.10.3 Expanding Analytical Modeling Approaches

The survey showed that evaluation within a user interface development environment (UIDE) is a promising approach for automated analysis via analytical modeling. Table 2.18 summarizes current approaches to analytical modeling. UIDE analysis methods, such as CRITIQUE and GLEAN, could be augmented with guidelines to support automated critique. Guidelines, such as thresholds for learning or executing certain types of tasks, could assist the designer with interpreting prediction results and improving UI designs. Evaluation within a UIDE should also make it possible to automatically optimize UI designs based on guidelines.

Although UIDE analysis is promising, it is not widely used in practice. This may be due to the fact that most tools are research systems and have not been incorporated into popular commercial tools. This is unfortunate since incorporating analytical modeling and possibly simulation methods within a UIDE should mitigate some barriers to their use, such as being too complex and time consuming to employ [Bellotti 1988]. Applying such analysis approaches outside of these user interface development environments is an open research problem.

Cognitive Task Analysis provides some insight for analyzing UIs outside of a UIDE. Furthermore, CTA is a promising approach for automated analysis, provided more effort is spent to fully develop this methodology. This approach is consistent with analytical modeling techniques employed outside of HCI, such as in the performance evaluation of computer systems [Jain 1991] (see Chapter 3); this is because with CTA the evaluator provides UI parameters to an underlying model for analysis versus developing a new model to assess each UI. However, one of the drawbacks of CTA is the need to describe the interface to the system. Integrating this approach into a UIDE or UIMS should make this approach more tenable.

As previously discussed, analytical modeling approaches for Web UIs still remain to be developed. It may not be possible to develop new approaches using a paradigm that requires explicit task hierarchies. However, a variation of CTA may be appropriate for Web UIs.

2.10.4 Expanding Simulation Approaches

Table 2.19 summarizes current approaches to simulation analysis. The survey showed that existing simulations based on a human information processor model have widely different uses (e.g., modeling a user interacting with a UI or solving a problem). 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 automated analysis, especially for evaluating alternative designs.

Method Class:	Analytical Modeling	
Automation Type:	Analysis	
Method Type:	UIDE Analysis - conduct GOMS analysis within a UIDE (4 methods)	
UE Method	**UI**	**Effort**
Generate predictions for GOMS task models (QGOMS, CATHCI)	WIMP	M
Generate GOMS task models and predictions (USAGE, CRITIQUE)	WIMP	M
Method Type:	Cognitive Task Analysis - predict usability problems (1 method)	
UE Method	**UI**	**Effort**
Cognitive Task Analysis (CTA)	WIMP	M
Method Type:	Programmable User Models - write program that acts like a user user (1 method)	
UE Method	**UI**	**Effort**
Programmable User Models (PUM)	WIMP	M

Table 2.18: Synopsis of automated analysis support for analytical modeling methods. This is a repetition of Table 2.11.

Method Class:	Simulation	
Automation Type:	Analysis	
Method Type:	Petri Net Modeling - mimic user interaction from usage data (1 method)	
UE Method	**UI**	**Effort**
Petri Net Modeling (AMME)	WIMP	IF
Method Type:	Information Processor Modeling - mimic user interaction (9 methods)	
UE Method	**UI**	**Effort**
Employ a computational cognitive architecture for UI analysis (ACT-R, COGNET, EPIC, HOS, Soar, CCT, ICS, GLEAN)	WIMP	M
Employ a GOMS-like model to analyze navigation (Site Profile)	Web	M

Table 2.19: Synopsis of automated analysis support for simulation methods. This is a repetition of Table 2.13.

It is possible to use several simulation techniques employed in the performance analysis of computer systems, in particular trace-driven discrete-event simulation and Monte Carlo simulation [Jain 1991], to enable designers to perform what-if analyses with UIs (see Chapter 3). Trace-driven discrete-event simulations use real usage data to model a system as it evolves over time. Analysts use this approach to simulate many aspects of computer systems, such as the processing subsystem, operating system, and various resource scheduling algorithms. In the user interface field, all surveyed approaches use discrete-event simulation. However, AMME constructs simulation models directly from logged usage, which is a form of trace-driven discrete-event simulation. Similarly, other simulators could be altered to process log files as input instead of explicit task or user models, potentially producing more realistic and accurate simulations.

Monte Carlo simulations enable an evaluator to model a system probabilistically (i.e., sampling from a probability distribution is used to determine what event occurs next). Monte Carlo simulation could contribute substantially to automated analysis by eliminating the need for explicit task hierarchies or user models. Most simulations in this domain rely on a single user model, typically an expert user. Monte Carlo simulation would enable designers to perform what-if analysis and study design alternatives with many user models. The approach employed by [Chi *et al.* 2000] to simulate Web site navigation is a close approximation to Monte Carlo simulation.

2.11 Summary

This chapter provided an overview of usability evaluation and presented a taxonomy for comparing various methods. It also presented an extensive survey of the use of automation in WIMP and Web interface evaluation, finding that automation is used in only 36% of methods surveyed. Of all of the surveyed methods, only 29% are free from requirements of formal or informal interface use. All approaches that do not require formal or informal use, with the exception of guideline review, are based on analytical modeling or simulation.

It is important to keep in mind that automation of usability evaluation does not capture important qualitative and subjective information (such as user preferences and misconceptions) that can only be unveiled via usability testing, heuristic evaluation, and other standard inquiry methods. Nevertheless, simulation and analytical modeling should be useful for helping designers choose among design alternatives before committing to expensive development costs.

Furthermore, evaluators could use automation in tandem with what are usually non-automated methods, such as heuristic evaluation and usability testing. For example, an evaluator doing a heuristic evaluation could observe automatically-generated usage traces executing within a UI.

Adding automation to usability evaluation has many potential benefits, including reducing the costs of non-automated methods, aiding in comparisons between alternative designs, and improving consistency in usability evaluation. Research to further develop analytical modeling, simulation, guideline review, and log file analysis techniques could result in several promising automated techniques. The next chapter discusses new approaches that could be developed based on performance evaluation of computer systems.

Chapter 3

New Automated Usability Evaluation Methods

3.1 Introduction

As discussed in Chapter 2, automated usability evaluation (AUE) methods are promising complements to non-automated methods, such as heuristic evaluation and usability testing. AUE methods enable an evaluator to identify potential usability problems quickly and inexpensively compared to non-automated methods and can decrease the overall cost of the evaluation phase [John and Kieras 1996; Nielsen 1993]. Despite the potential benefits of AUE methods, this field is greatly underexplored as shown by the survey in Chapter 2.

Performance evaluation (PE) encompasses established methodologies for measuring the performance (e.g., speed, throughput, and response time) of a system and for understanding the cause of measured performance [Jain 1991]. Since computer systems were first invented in the 1950's, system designers and analysts have used these methodologies extensively to compare and improve the performance of hardware and software systems.

The intent of this chapter is to illustrate how PE provides insight about new methods for automated usability assessment. As such, the chapter systematically compares the two methodologies. It provides background for PE, discusses the mapping between the two methodologies, and introduces two example applications. It then describes how to apply PE to UE for three different classes of evaluation: measurement, simulation, and analytical modeling. In each case, the discussion illustrates the potential for new automated usability evaluation methods based on this comparison.

3.2 The Performance Evaluation Process

System designers, analysts, and high performance computing experts have used performance evaluation techniques extensively to improve and compare the performance of hardware and software systems. More recently, human performance and process engineers began to use these methodologies to understand and improve the performance of humans and work practices. Performance in these contexts is a measure of the speed at which a system (e.g., a computer, person, or process) operates and/or its total effectiveness, including throughput, response time, availability, and reliability. Performance evaluation encompasses methodologies for measuring and for understanding the cause of measured performance. Although this methodology has been utilized in many domains, this chapter focuses on its application in the computer hardware and software domain.

1. Specify performance evaluation goals.
2. Define system boundary.
3. List system services and outcomes.
4. Select performance metrics.
5. List system and workload parameters.
6. Select factors and their levels.
7. Select evaluation method(s).
8. Select workload.
9. Implement evaluation program.
10. Design experiments.
11. Capture performance data.
12. Analyze and interpret performance data.
13. Critique system to suggest improvements.
14. Iterate the process if necessary.
15. Present results.

Figure 3.1: Activities that may occur during the performance evaluation process.

Performance evaluation is a process that entails many activities, including determining performance metrics, measuring performance, and analyzing measured performance. Jain [1991] and Law and Kelton [1991] present similar ten-step systematic approaches to performance evaluation; these steps were adapted for this discussion. Figure 3.1 depicts a fifteen-step process for conducting a performance evaluation. This process is similar to the one outlined for usability evaluation in Chapter 2. Jain [1991] and Law and Kelton [1991] provide detailed discussions of the steps comprising the performance evaluation process. Hence, they are not discussed in this chapter.

3.3 Overview of Performance Evaluation Methods

PE consists of three broad classes of evaluation methods: measurement, simulation, and analytical modeling [Jain 1991]. A key feature of all of these approaches is that they enable an analyst to automatically generate quantitative performance data. Such data can: (i) help designers explore design alternatives; (ii) help analysts tune system performance; and (iii) help consumers purchase systems that satisfy their performance requirements. The methods differ along several dimensions, including the system stage at which they are applicable, cost, accuracy, time, and resource requirements. Table 3.1 compares the methods along these dimensions; the table is modeled after the comparison in [Jain 1991]. The columns reflect the order of importance of dimensions.

Measurement has the potential to be the most accurate and credible evaluation method; however, it is only applicable to an existing system – either the target system or one similar to it. Simulation is usually more accurate and credible than analytical modeling, since it allows

Method	Stage	Time	Resources
Analytical Modeling	Any	Small	Analysts
Simulation	Any	Medium	Computer Languages
Measurement	Post-prototype	Varies	Instruments

Method	Accuracy	Trade-off	Cost	Credibility
Analytical Modeling	Low	Easy	Small	Low
Simulation	Moderate	Moderate	Medium	Medium
Measurement	Varies	Difficult	High	High

Table 3.1: Summary of the performance evaluation method classes.

the analyst to incorporate greater system detail. However, simulations require considerably more time to develop than analytical models. Analytical modeling is usually the least accurate and consequently the least credible evaluation method because of simplifications and assumptions made to produce the model. However, this method can be valuable for comparing alternatives during early system design.

Due to the various trade-offs associated with each evaluation method, it is advisable to employ two or three methods simultaneously [Jain 1991; Law and Kelton 1991; Sauer and Chandy 1981]. Typically, an analyst may use simulation and analytical modeling together to verify and validate the results of each one. It is especially advisable to use analytical modeling and/or simulation along with measurement, since measurement is highly susceptible to experimental errors.

The remaining sections discuss analytical modeling, simulation, and measurement methods in more detail.

3.4 Measurement Methods

Capturing real system performance is by far the most credible, yet most expensive performance evaluation method. It requires careful selection of performance metrics, a means of running a workload, and a means of capturing performance measures for the workload. The following sections discuss these aspects.

3.4.1 Measurement Methods: Selecting Performance Metrics

Performance metrics are a crucial component of this type of assessment and care must be taken when selecting measures. The goal is to choose a minimal number of metrics that reveal the maximum amount of relevant performance detail for the system under study. In instances where multiple users share the system under evaluation (e.g., distributed computing), the analyst must give consideration to both individual and global metrics. Individual metrics reflect performance for each user, while global metrics reflect the system-wide performance.

Jain [1991] suggests using the list of possible service outcomes (correct response[1], incorrect response, or nonresponse) to guide the selection of performance metrics. Jain distinguishes three categories of metrics – speed, reliability, and availability – corresponding to the three possible outcomes. Jain also distinguishes between metrics that measure individual (single user) and global

[1]Response is a generic term applicable to a wide range of services (e.g., query processing, I/O processing, CPU processing, etc.).

Outcome	Metric Type	Example Metric	Scope	Performance Goal
Correct Response	Speed	response time	IG	LB
		throughput	IG	HB
		utilization	G	NB
Incorrect Response	Reliability	probability of error	G	LB
		time between errors	G	LB
Nonresponse	Availability	probability of failure	G	LB
		time between failures	G	LB

Table 3.2: Metrics associated with possible outcomes. Reliability and availability metrics are typically captured over a time period (e.g., weeks or months) and are used to assess system performance across users versus for individual users. The scope of a metric is an individual user (I), global or across users (G), or both. Performance goals include: lower is better (LB), higher is better (HB), and nominal or middle is better (NB).

(across users) performance. Table 3.2 summarizes these distinctions along with example metrics. The scope of each metric – individual or global – is denoted with an I and G, respectively. The following performance goals are also appropriately associated with each example metric: LB - lower is better; HB - higher is better; and NB - nominal or middle is better. Below is a discussion of each of the metric types.

- **Speed:** a measure of system response time, throughput, or utilization. Response time is the time between a user's request and the system's response to the request. Throughput is the rate (in requests per unit of time) at which the system services requests. Utilization is the fraction of time the system is busy servicing requests.
- **Reliability:** a measure of the fraction of time the system correctly services users' requests.
- **Availability:** a measure of the fraction of time the system is available to service user requests.

It is possible that an analyst may need to employ several metrics, individual and global, for each system service; hence, the number of metrics can grow proportionally (e.g., twice the number of services to be evaluated). To reduce the number of metrics, Jain suggests selecting a subset of metrics with low variability, nonredundancy, and completeness as discussed below.

- **Low Variability:** metrics that are not a ratio of two or more variables.
- **Nonredundancy:** metrics that do not convey essentially the same information.
- **Completeness:** metrics that reflect all of the possible outcomes of a service.

3.4.2 Measurement Methods: Running a Workload

In order to capture performance data, the analyst must first load a workload onto the system via a load driver. Internal drivers, live operators, and remote terminal emulators are the major types of load drivers and are discussed below.

- **Internal Driver:** encapsulates mechanisms for loading and running the workload into one program, as is the case in benchmarks. One problem with this approach is that loading the workload may affect system performance during execution. Nonetheless, benchmarks are a commonly used internal load driver. They are discussed in more detail below.

- **Live Operators:** real users submit requests to the system. The use of live operators is extremely costly and hard to control.

- **Remote Terminal Emulators:** programs that run on separate computers and submit requests to the system under study. This is one of the most popular and desirable load drivers used; it is not nearly as costly as using live operators, and it eliminates the interference problem of internal drivers.

Measurement Methods: Running a Workload – Benchmarking

Benchmarking [Dowd 1993; Jain 1991; Weicker 1990], is one of the most commonly used measurement techniques for comparing two or more systems. Benchmarking entails using a high-level, portable program with a realistic workload and adequate problem size to automatically quantify performance aspects in a reproducible manner. A single such program is referred to as a benchmark, while multiple programs used jointly are referred to as a benchmark suite. Benchmarks automatically specify performance metrics and workloads. As such, they simplify the performance evaluation process to some degree.

Benchmarks differ along several dimensions, including the application domain, the code type used, and the type of execution measured as summarized below.

- **Application Domain:** the benchmark workload and problem size typifies a specific application domain (e.g., scientific computing, graphics, or databases) and measures performance aspects germane to this domain. For example, a scientific computing benchmark may assess floating-point computation speed, whereas a database benchmark may assess transaction completion speed or throughput.

- **Code Type:** the actual code executed and measured can be a real application (i.e., it exercises system resources as fully and realistically as possible), a kernel (i.e., the computation "core" of an application), or a synthetic program designed to model activity of a typical program or to mimic a real workload.

- **Execution Type:** the benchmark code can measure several execution types - single stream, throughput, or interactive. Single stream benchmarking measures the time to execute one or more benchmarks individually. Throughput benchmarking collects measurements while multiple programs run concurrently. Use of the term throughput here is distinct from its use as a performance measure by both single stream and interactive benchmarks. In these instances multiple programs are not running concurrently; hence, they are not considered throughput benchmarks. Throughput benchmarks are not as common nowadays as they were with batch processing systems. Interactive benchmarking measures response time or throughput in a client/server environment; this execution type requires a second program to simulate user requests to the server.

Several industry benchmarks provide objective measures of system performance. Standard benchmarks help buyers to make informed purchases and vendors to prioritize optimization in their systems. One drawback of these benchmarks is that they lack orthogonality because they sometimes measure many things at once; this makes it difficult to draw concrete conclusions about performance. Table 3.3 summarizes several commonly used industry benchmarks. There are published performance results from a myriad of systems for each of these benchmarks; thus, enabling comparison across systems.

Benchmark	Application Domain	Code Type	Execution Type
Linpack	scientific computing	application	single stream
STREAM	scientific computing	synthetic	single stream
TPC-C	database	synthetic	interactive
TPC-D	database	synthetic	interactive
SPEC Web 96	Web	synthetic	interactive

Benchmark	What's Measured	How It's Measured	Metric
Linpack	floating point speed	solving linear equation	MFlops
STREAM	memory bandwidth	performing vector operations	MB/s
TPC-C	server throughput	servicing complex transactions	tpmC
TPC-D	server throughput	servicing complex db queries	QphD
SPEC Web 96	server throughput	servicing HTTP GET requests	SPECweb96

Table 3.3: Summary of commonly used industry benchmarks. The What's Measured column reflects the performance aspect measured by each benchmark. The How It's Measured column describes the workload used for measuring this performance aspect, while the Metric column lists the measurement returned by each benchmark.

3.4.3 Measurement Methods: Capturing Performance Metrics

Analysts use several types of monitors (e.g., hardware counters and software timers) as well as accounting logs to capture performance data as discussed below.

- **Monitors:** measure system performance at some level. Hardware monitors, such as counters, measure low-level performance aspects (e.g., signals on buses and instruction execution time). Software monitors, such as code instrumented with timing calls, measure high-level performance aspects (e.g., queue lengths and time to execute a block of code). There are also firmware monitors, such as a processor microcode instrumented with timing calls, that measure performance aspects of network interface cards and other external system components. Sometimes analysts use multiple hardware, software and/or firmware monitors simultaneously as a hybrid monitor.

- **Accounting Logs:** another form of software monitors that automatically capture performance data during program execution. Hence, they do not require system instrumentation as is the case for monitors. Usually, compiling a program with certain flags enables accounting.

Tracing and sampling are the primary measurement techniques used in monitors and accounting logs. Tracing actually produces a time-stamped record of requests as they move through various stages of the system and/or of event occurrences in the system. Sampling or timing entails reading a clock at specified times to compute elapsed time between timing events.

3.5 Analytical Modeling Methods

An analytical model consists of mathematical or logical relationships that represent the analyst's assumptions about how a system works. By solving this model, the analyst can predict system performance. Analytical modeling approaches range in complexity from simple "back of the envelope" calculations to formal queuing theory as discussed below.

Simplistic Models. "Back of the envelope" and "front of the terminal" [Sauer and Chandy 1981] calculations are two relatively simple analytical modeling approaches. In the "back of the envelope" approach the system model is extremely crude such that it facilitates solving the model via unaided calculation. An analyst may solve slightly more complicated models using modeling software or a simple mathematical environment, such as Matlab; this is considered to be "front of the terminal" calculation. Since simplistic models are very abstract representations of systems, the accuracy of the results produced are highly questionable. Nonetheless, this approach is very appropriate during the design stage of a system.

Informal Queuing Theory. In this approach an analyst develops a queuing model (i.e., a model of the times a request spends in various system resource queues) that captures all of the significant aspects of the system. The analyst then uses this model to get an approximation (e.g., using numerical methods) to the model solution (i.e., response time). The model is usually significantly more detailed than the previous approach but less thorough and accurate than formal queuing theory. The latter case is due to the use of approximate versus exact system parameters.

Formal Queuing Theory. Formal queuing theory requires more accurate specification of the following six system parameters: interarrival times of requests; service time at each queue; number of resources or servers; system capacity (i.e., number of requests that can be serviced); population size or the maximum number of requests; and the service discipline or policy for servicing requests, such as first come first served. This approach can also be used to model multiple queues in a system as a queuing network. The objective of formal queuing theory is to solve models for parameters that impact performance to answer questions, such as the number of servers required to fulfill the current demand or appropriate sizes of queue buffers to prevent overflow.

3.6 Simulation Methods

Some models may be too complex to solve using analytical modeling or may have no analytical solution. In these cases the analyst can create a simulation (i.e., a computer program) to exercise the model with various inputs to see the resulting affects on output measures of performance. The analyst accomplishes this with a system model, a program or simulator that behaves like the model, and detailed input data to the program. Simulation allows the analyst to create arbitrarily detailed models of systems. Consequently, performance analysts consider simulation to be more credible and accurate than analytical modeling. Nonetheless, simulation requires considerably more time to develop as well as compute resources to run.

One of the most important aspects of a simulation is the underlying simulation model. Simulation models vary along three major dimensions as discussed below.

- **System Evolution:** time-independent models are a representation of a system at a particular time that is totally independent of time, whereas time-dependent models represent a system as it evolves over time.
- **System Parameters:** deterministic models do not contain probabilistic (i.e., generated based on random numbers) input components, while probabilistic models may contain some random input components.
- **Number of System States:** finite models have a countable number of system states, whereas infinite models have an uncountable number of states.

These aspects of the model largely govern the type of simulation the analyst uses during in performance studies. Below is a discussion of several commonly used simulation approaches.

Discrete-Event. In a discrete-event simulation the state variables of the system change instantaneously in response to events.

Continuous-Event. In this simulation approach the state variables of a system change continuously with respect to time.

Combined Discrete-Continuous. A simulation can use both the discrete-event and continuous-event approaches. For example, a discrete system change may occur when a continuous variable reaches a threshold value.

Monte Carlo. A Monte Carlo simulation uses random numbers to solve stochastic or deterministic models without time dependence. Analysts use this simulation approach to model probabilistic phenomenon that do not change characteristics with time, such as cancer cell growth.

Any of these simulation approaches, with the exception of Monte Carlo simulation, can be driven by a time-ordered record of events taken from a real system. This is referred to as trace-driven simulation. Analysts consider this type of simulation to be the most credible and accurate.

Several problems associated with all of these simulation approaches include: an inappropriate level of system detail, poor random number generators and seeds, no verification or validation of models, too short simulation runs, and too long simulation runs due to the model complexity.

3.7 Mapping Between Performance and Usability Evaluation

As previously discussed, performance evaluation encompasses established methodologies for measuring the performance (e.g., speed, throughput, and response time) of a system and for understanding the cause of measured performance [Jain 1991]. PE consists of three broad classes of evaluation methods: measurement, analytical modeling, and simulation. A key feature of all of these approaches is that they enable an analyst to automatically generate quantitative performance data.

Usability evaluation, on the other hand, encompasses methodologies for measuring usability aspects (e.g., effectiveness, efficiency, and satisfaction) of an interface and for identifying specific problems [Nielsen 1993]. As discussed in Chapter 2, UE consists of five classes of methods: testing, inspection, inquiry, analytical modeling, and simulation. Although there has been some work to automate these approaches, automated UE methods are greatly underexplored. Furthermore, only 29% of existing methods support the same level of automation as PE methods (i.e., automated capture, analysis, or critique without requiring interface usage).

The remainder of this chapter discusses how PE can be used as a guiding framework for developing new automated UE methods for both WIMP (Windows, Icons, Menus and Pointers) and Web interfaces. Similarly to PE, automated UE methods can provide additional support to evaluators using non-automated methods and for designers exploring design alternatives. These methods are even more crucial in instances where performance is a major consideration.

Sections 2.3 and 3.2 show the UE and PE processes to be quite similar. Furthermore, there is a mapping between the methods used within each domain. Table 3.4 summarizes this mapping. PE measurement, analytical modeling, and simulation methods are used as a framework for this discussion. The following section presents two applications used throughout the comparison of PE and UE.

PE	UE
measurement	testing, inspection, inquiry
analytical modeling	analytical modeling
simulation	simulation

Table 3.4: Mapping between performance evaluation (PE) and usability evaluation (UE) method classes.

Figure 3.2: Address label example in Microsoft Word.

3.7.1 Example Applications and Tasks

Two example applications and representative tasks are referred to throughout this discussion. The first application is a word processor with a task of creating address labels – six to a page where each label has a rectangular border. Figure 3.2 depicts a sample set of address labels in Microsoft Word.

It is assumed that the user loaded the application prior to beginning the task and that the cursor is at the top left margin of a blank document when the user begins the label creation task. Analysis of this task within Microsoft Word 97 revealed that it requires an expert user 55 steps to complete (see Appendix B). A replicated analysis in Microsoft Word 2000 derived the same number of steps. Figure 3.3 shows the first thirteen steps required to create the first label, along with the corresponding high-level goals; Appendix B contains the high-level goals for all 55 steps in the task sequence. A high-level task structure similar to NGOMSL [John and Kieras 1996] is used in discussions.

The second application is an information-centric Web site typical of large-scale federal and state agencies. The task for this application is to navigate from a site entry point to a page that contains some target information. Figure 3.4 depicts this example with a sample navigation path. Unlike the label creation example, there is no clear step-by-step procedure for completing the task. The user could start from any page within the site and follow various paths to the target information. Hence, it is not possible to specify an explicit task structure as specified for the label creation example without restricting users to traversing one navigation path through the site. It is assumed that users are unfamiliar with the site and that locating the information is a one-time

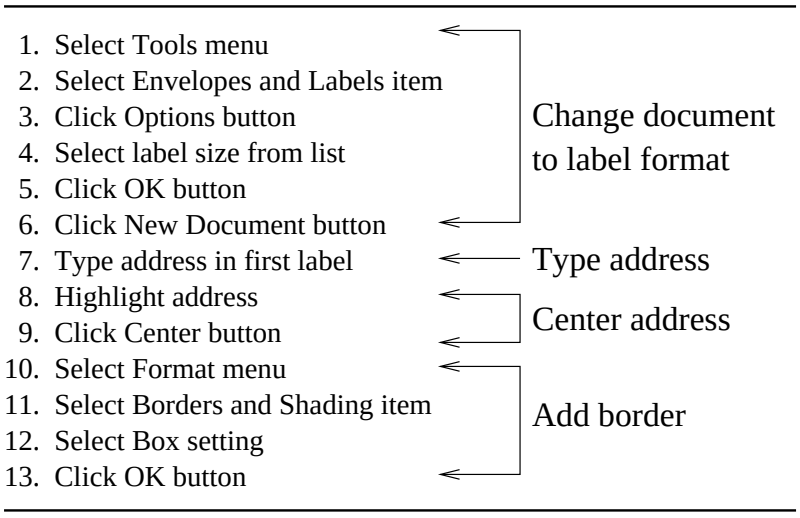


Figure 3.3: Label creation steps in Microsoft Word 97 and 2000.

Measurement Workload	**Measurement Type**	
	Monitoring	Profiling
Benchmark		
Granularity	fine	coarse
Interference	yes	yes
Real User		
Granularity	fine	coarse
Interference	no	no

Table 3.5: Characteristics of measurement techniques in the performance evaluation domain.

task. Thus, bookmarking is not used to access the information. It also assumed that users enter the site via a search engine or external link.

The remainder of this chapter demonstrates how to determine the number of errors and the navigation time for the label creation and site navigation tasks, respectively.

3.8 New UE Measurement Methods

3.8.1 Measurement in Performance Evaluation

Measurement is by far the most credible, yet most expensive PE method [Jain 1991]. It requires a means of running a workload on a system as well as a means of capturing quantitative performance data while the workload runs. A performance analyst usually derives the workload from current or anticipated system use. Capturing quantitative data for the workload enables the analyst to identify performance bottlenecks, tune system performance, and forecast future performance. Table 3.5 summarizes relevant techniques for running workloads and capturing quantitative performance data. Section 3.4 discusses these approaches in more detail.

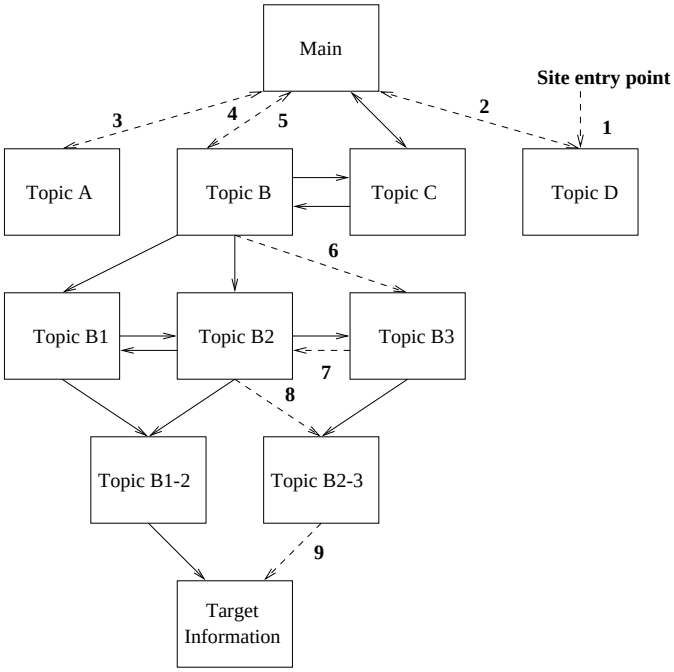


Figure 3.4: Information-centric Web site example.

3.8.2 Measurement in Usability Evaluation

Inspection, testing, and inquiry UE methods are equivalent to PE measurement techniques; most require an interface or prototype to exist to capture measurements of some form (e.g., number of heuristic violations, task completion time, and subjective rating). Most evaluation methods surveyed in Chapter 2 do not produce quantitative data. However, methods that do produce quantitative data, such as performance measurement[2], use both profiling and monitoring techniques (e.g., high-level and low-level logging) similarly to their PE counterparts, and all of these methods require a real user (an evaluator or test participant).

Although monitoring with benchmarks is the predominate approach in the PE domain, it is unused in the UE domain. The closest approximation is replaying previously-captured usage traces in an interface [Neal and Simons 1983; Zettlemoyer *et al.* 1999]. Early work with Playback [Neal and Simons 1983] involved recording actions performed on the keyboard during usability testing and then sending the recorded commands back to the application. The evaluator could then observe and analyze the recorded interaction.

More recent work automatically generates usage traces to drive replay tools for Motif-based UIs [Kasik and George 1996]. The goal of this work is to use a small number of input parameters to inexpensively generate a large number of test scripts that a tester can then use to find weak spots and application failures during the design phase. The authors implemented a prototype system that enables a designer to generate an expert user test script and then insert deviation commands at different points within the script. The system uses a genetic algorithm to choose user behavior during the deviation points as a means for simulating a novice user learning by experimentation.

Recent work in agent technology captures widget-level usage traces and automatically

[2]Performance measurement in this context refers to usability testing methods rather than the performance evaluation method.

analyzes user actions during replay [Zettlemoyer *et al.* 1999]. The IBOT system interacts with Windows operating systems to capture low-level window events (e.g., keyboard and mouse actions) and screen buffer information (i.e., a screen image that can be processed to automatically identify widgets). The system then combines this information into interface abstractions (e.g., menu select and menubar search operations) that it can use to infer UI activities at a high-level. The IBOT system can also perform the same operations as a real user by adding window events to the system event queue. Similar work has been done in the software engineering field for generating test data values from source code [Jasper *et al.* 1994].

Guideline review based on quantitative measures is a somewhat distant approximation of PE benchmarking. Automated analysis tools, such as AIDE (semi-Automated Interface Designer and Evaluator) [Sears 1995], compute quantitative measures for WIMP UIs and compare them to validated thresholds. Similar, underdeveloped approaches exist for Web UI assessment (e.g., HyperAT [Theng and Marsden 1998], Gentler [Thimbleby 1997], and the Rating Game [Stein 1997]). Thresholds either do not exist or have not been empirically validated.

3.8.3 Applying PE Measurement Methods to UE

As previously stated, benchmarking is widely used in PE to automatically capture quantitative performance data on computer systems. Usability benchmarks, especially benchmarks that can be executed within any UI, is a promising open area of research. Nonetheless, there are three major challenges to making this a reality in the UE domain. Brief discussions of these challenges and potential solutions are below; the next section provides more depth on these issues.

The first challenge is generating usage traces without conducting usability testing or reusing traces generated during usability testing. One approach was discussed above [Neal and Simons 1983]; however, more work needs to be done to automatically generate traces that represent a wider range of users and tasks to complement real traces. In particular, a genetic algorithm could simulate usage styles other than a novice user learning-by-experimentation [Kasik and George 1996]. It may also be beneficial to implement a hybrid trace generation scheme wherein traditional random number generation is used to explore the outer limits of a UI [Kasik and George 1996]. Data from real users, such as typical errors and their frequencies, could serve as input in both cases to maximize the realism of generated tasks.

The second challenge is making such traces portable to any UI. In the PE domain this is accomplished by writing hardware-independent programs in a common programming language, such as C or Fortran. Analysts then compile these programs with the appropriate compiler flags and execute them to capture performance data. There needs to be a similar means of creating portable usage traces. One way may entail mapping high-level tasks captured in a trace into specific interface operations. The USINE system [Lecerof and Paternò 1998] provides insight on accomplishing this. In this system, evaluators create task models expressing temporal relationships between steps and create a table specifying mappings between log file entries and the task model. USINE uses this information to identify task sequences in log files that violate temporal relationships. Another approach employed in the IBOT system is to use system-level calls (e.g., mouse and keyboard event messages); this simplifies porting to other operating systems. The authors claim that the IBOT agent can interact with any off-the-shelf application because it is independent of and external to the UI.

A proposed solution would combine both the USINE and IBOT approaches. Similarly to USINE, a task programming tool could prompt evaluators for application steps corresponding to each task within a generated trace. Ideally, the evaluator could use programming-by-demonstration [Myers 1992] to record application steps. The task programming tool could translate application

Challenge	PE	UE
Executable Workload	software programs	usage traces (real & generated)
Portability	hardware-independent software programs	high-level traces with UI mapping, benchmark program generation
Quantitative Metrics	execution time, standard metrics	number of errors, navigation time

Table 3.6: Summary of the challenges in using PE measurement techniques within the UE domain. The PE column describes how these challenges are resolved within the PE domain, and the UE column describes potential ways to resolve these challenges within the UE domain.

sequences into a format, such as system-level calls, that could be subsequently executed within the application. A benchmark generation tool could then translate the usage trace and mapped application sequences into a program for execution similarly to replay tools; each application sequence could be expressed as a separate function in the program. The Java programming language is promising for the task programming tool, the benchmark generation tool, as well as the generated benchmark programs.

The final challenge is finding a good set of quantitative metrics to capture while executing a trace. Task completion time, the number and types of errors, and other typical UI metrics may suffice for this. One of the drawbacks of relying solely on quantitative metrics is that they do not capture subjective information, such as user preferences given a choice of UIs with comparable performance and features.

Usability benchmarks are appropriate for evaluating existing UIs or working prototypes (as opposed to designs). Evaluators can use benchmark results to facilitate identifying potential usability problems in two ways: (i) To compare the results of an expert user trace to results from those generated by a trace generation program. This may illustrate potential design improvements to mitigate performance bottlenecks, decrease the occurrence of errors, and reduce task completion time. (ii) To compare results to those reported for comparable UIs or alternative designs. This is useful for competitive analysis and for studying design tradeoffs. Both of these uses are consistent with benchmark analysis in the PE domain.

Table 3.6 summarizes the challenges with using benchmarking in the UE area. The next section describes usability benchmarks for the example tasks in more detail.

3.8.4 Example Usability Benchmark

As previously discussed, executable and portable usage traces and a set of quantitative performance metrics are required to construct a benchmark. To generate high-level usage traces, the evaluator could specify a high-level representation of the label creation task sequence previously discussed. Figure 3.5 depicts the nine high-level steps that correspond to the 55-step Microsoft Word sequence. It is also possible to process a real user trace to create a high-level task sequence.

Figure 3.6 demonstrates a procedure for using the high-level task sequence as input for constructing and executing a usability benchmark. The evaluator could use the high-level trace as a task template in which deviation points could be identified similarly to the work done in [Kasik and George 1996]. The trace generation program would then generate plausible variations of the task sequence that represent alternative user behaviors during task completion. Figure 3.7 shows the type of output that the program might generate – an example in which a user mistakenly enters text before changing the document format, corrects the mistake, and completes the task as specified by the task template. This approach enables the evaluator to amplify the results taken

1. Change document to label format
2. Enter text
3. Center text
4. Add square border to text
5. Copy text
6. Move cursor to next label
7. Paste text
8. Add square border to text
9. Repeat steps 6-8 for remaining 4 labels

Figure 3.5: High-level steps for the label creation task.

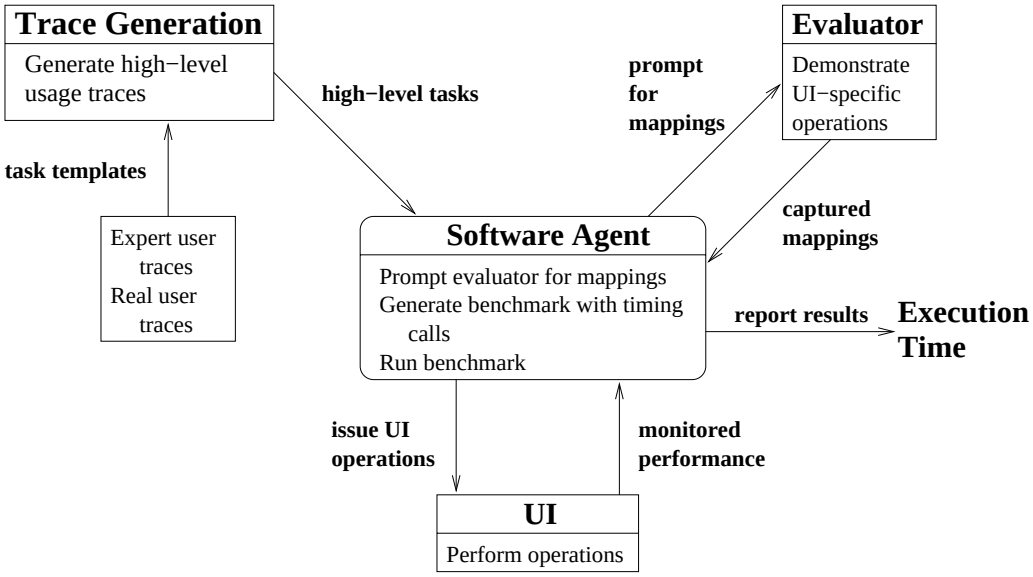


Figure 3.6: The proposed usability benchmarking procedure.

from a small number of inputs or test subjects to generate behavior equivalent to a larger number of users and wider UI coverage.

An evaluator or designer could then map all of the high-level tasks in the generated traces into equivalent UI-specific operations as depicted in Figure 3.6. Figure 3.3 shows one such mapping for creating the first label in Microsoft Word. Programming-by-demonstration [Myers 1992] is one way to facilitate this mapping. An agent program could prompt the designer to demonstrate a task sequence, such as delete document, and record the steps to facilitate playback. If there is more than one way to accomplish a task, then the designer could demonstrate multiple methods for a task and specify a frequency for each, similarly to GOMS [John and Kieras 1996] task representations.

Given the high-level usage traces and UI mappings, the agent could then automatically generate executable benchmark programs to replay in a UI as depicted in Figure 3.6. To facilitate error analysis, the agent could also generate a designer benchmark (i.e., a benchmark representing the correct sequences of operations to complete a task) from the task template and UI mappings. The agent would then "run" (i.e., send operations to the UI) this designer benchmark once to record

1. Enter text
2. Change document to label format
3. Delete document
4. Create new document
5. Change document to label format
6. Enter text

⋮

Figure 3.7: Example trace for the label creation task.

1. Select Topic D page (Site entry point)
2. Select Main link
3. Select Topic A link
4. Click Back button
5. Select Topic B link
6. Select Topic B3 link
7. Select Topic B2 link
8. Select Topic B2-3 link
9. Select Target Information link

Figure 3.8: Example trace for the information-seeking task.

system state after each operation. The agent would repeat this procedure for each generated benchmark, record discrepancies (i.e., differences in resulting system state between the designer benchmark and other benchmarks) as errors, note error locations, and report whether the task completed successfully[3]. The agent could also aggregate data over multiple traces to provide more conclusive error analysis data.

Similar approaches using generated benchmark programs as well as quantitative measures could be applied to the information-centric Web site example. Generating traces for multiple navigation paths (and in some cases all possible paths) is the most crucial component for this example. An algorithm can determine plausible paths based on the navigation structure and content of the links and pages. Chi *et al.* [2000] also demonstrates a methodology for generating plausible navigation paths based on information scent. Genetic algorithm modeling could also be employed for this. Again, as input the evaluator could use real user traces from Web server logs or a designer-generated trace. Since navigation operations (e.g., select a link, click the back or forward button, etc.) are standard in Web browsers, it may be possible to eliminate the mapping operation that was required for the WIMP example. Hence, the genetic algorithm could generate executable benchmarks directly as depicted in Figure 3.8. Figure 3.4 shows the corresponding navigation path.

A software agent could simulate navigation, reading, form completion, and other user behavior within the actual site and report navigation timing data. WebCriteria's Site Profile tool [Web Criteria 1999] uses a similar approach to simulate a user's information-seeking behavior within a model of an implemented Web site. Site Profile uses a standard Web user model to follow an

[3] Actually "running" both benchmarks in a UI is required, since it may not be possible to ascertain successful task completion by comparing two task sequences. This can only be accomplished by comparing system state.

explicit navigation path through the site and computes an accessibility metric based on predictions of load time and other performance measures. This approach suffers from two major limitations: it uses a single user model; and it requires specification of an explicit navigation path. The proposed approach with automatically-generated navigation paths would not have these limitations.

Another benchmarking approach for Web sites could entail computing quantitative measures for Web pages and sites and comparing these measures to validated thresholds or profiles of highly-rated sites. As previously discussed, HyperAt, Gentler, and the Rating Game compute quantitative measures (e.g., number of links, number of words, and breadth and depth of each page), but validated thresholds have not been established. This dissertation presents an empirical framework for using quantitative measures to develop profiles of highly-rated sites. Such profiles could also be used to determine validated thresholds. Subsequent chapters discuss the methodology, profiles, and threshold derivations in more detail.

3.9 New UE Analytical Modeling Methods

3.9.1 Analytical Modeling in Performance Evaluation

Analytical modeling entails building mathematical or logical relationships to describe how an existing or proposed system works. The analyst solves a model to predict system performance. Such predictions are useful for studying design alternatives and for tuning system performance in the same manner as measurement and simulation. The major difference is that analytical modeling is much cheaper and faster to use albeit not as credible [Jain 1991].

Most analytical models do not adhere to a formal framework, such as queuing theory (see Section 3.5). These models use parameterized workloads (e.g., request characteristics determined by probability distributions) and vary in complexity. Some models may be solved with simple calculations, while others may require the use of software.

3.9.2 Analytical Modeling in Usability Evaluation

The survey in Chapter 2 revealed analytical modeling methods for WIMP UIs, but not for Web interfaces. GOMS analysis [John and Kieras 1996] is one of the most widely-used analytical modeling approaches, but there are two major drawbacks of GOMS and other UE analytical modeling approaches: (i) They employ a single user model, typically an expert user, and (ii) They require clearly-defined tasks. The latter is appropriate for WIMP UIs but does not work well for information-centric Web sites for reasons previously discussed.

One way to address the first problem is to construct tasks representative of non-expert users with a programming-by-demonstration facility embedded within a UIDE. CRITIQUE [Hudson *et al.* 1999] is one such tool; it automatically generates a GOMS structure for a task demonstrated within a UIDE. Constructing tasks representative of non-expert users requires the evaluator or designer to anticipate actions of novice and other types of users. However, this information is usually only discovered during usability testing. Thus, it has a strong non-automated component.

3.9.3 Applying PE Analytical Modeling to UE

Another approach to address both of the problems discussed above can be derived from the PE analytical modeling framework. Recall from Section 3.3 that PE analytical models predict system performance based on input parameters. To study different usage scenarios, the analyst simply changes the input parameters, not the underlying system model. Analytical modeling in the

Challenge	PE	UE
Modeling System	parameterized model	parameterized model
Multiple Usage Scenarios	vary input parameters	usage traces (real & generated), vary input parameters

Table 3.7: Summary of the challenges in using PE analytical modeling techniques in the UE domain. The PE column describes how these challenges are resolved within the PE domain, and the UE column describes potential ways to resolve these challenges within the UE domain.

UE domain is usually performed in a manner contrary to the PE counterpart (especially techniques using GOMS). These approaches require the evaluator to change the underlying system model rather than the input parameters to study different usage scenarios. Below is a brief discussion of challenges and potential solutions for addressing this problem; the next section presents a more in depth solution.

It would be better to construct an abstracted model (i.e., no task details) of a UI that encapsulates the most important system parameters for predicting performance (e.g., baseline time to enter information in a dialog box or scan a Web page for novice and expert users). For example, the designer could construct a basic graphical UI model to predict the number of errors for a high-level task sequence based on the number of operations in a task, the probability of errors, and other relevant interface parameters. The designer could then vary the input parameters (e.g., the number of dialog boxes required by a task and that predictions are to be based on a novice user) to the model. Each variation of input parameters corresponds to a different design or different user model. This allows for quick, coarse comparison of alternative designs. Previous work on GOMS and usability studies could be used to construct this basic UI model. Such models inherently support ill-defined task sequences, since they only require specification of key parameters for tasks. Although predictions from the model would be crude, such predictions have proven to be invaluable for making design decisions in PE [Jain 1991].

Besides constructing an abstract model, the designer must determine appropriate system and input parameters. If an interface exists, either as an implemented system or a model, then the designer could process generated or real usage traces to abstract the required input parameters. If an interface or interface model does not exist, then the designer must specify required input parameters manually.

Cognitive Task Analysis (CTA) [May and Barnard 1994] employs a modeling approach that is similar to the one proposed. CTA requires the evaluator to input an interface description to an underlying theoretical model for analysis. The theoretical model, an expert system based on Interacting Cognitive Subsystems (ICS [Barnard 1987]; discussed in Section 2.9), generates predictions about performance and usability problems similarly to a cognitive walkthrough. The CTA system prompts the evaluator for interface details from which it generates predictions and a report detailing the theoretical basis of predictions. Users have reported experiencing difficulties with developing interface descriptions. An approach based on quantitative input parameters should simplify this process.

Table 3.7 summarizes the challenges for using analytical modeling in the UE domain as it is used in PE. Analytical modeling is most appropriate for performing quick, coarse evaluations of various design alternatives.

3.9.4 Example Analytical Models

Analytical modeling is appropriate for comparing high-level designs in order to inform design decisions. Below are example comparison scenarios for the label creation and Web site navigation tasks.

Label Creation: A designer wants to develop a wizard for walking users through the cumbersome label creation process. The designer is considering one wizard design that has only three steps (dialog boxes) but requires the user to specify multiple things (e.g., various label and page settings) during each step. The designer is also considering a design with five steps wherein the user specifies fewer things during each step. The designer wants to know which of the designs would be easier to use.

Web Site Navigation: A designer needs to determine how to divide some content over multiple pages. The designer is considering dividing the content over five large pages as well as over eight medium pages. The designer wants to know which of the designs would be easier to navigate, especially for first-time visitors.

Equation 3.1 demonstrates a way to predict task completion time; this calculation could be embedded within a simple UI model and used to generate predictions for the label creation wizard. This model could contain baseline or average times for completing various interface operations (*task_type*), such as entering information in a dialog box or performing a generic task. The model could also incorporate high-level task complexity (*complexity_adj*) and user (*user_type*) models. For example, if the designer specified prediction based on a novice user, then the model could adjust the baseline time for tasks (e.g., increase by 15%). Similarly, if the designer specified that tasks were highly complex, then the model could further adjust the baseline time. To generate predictions, the designer need only specify the task type (dialog task), the task complexity (medium for the first design and low for the second), and the user type (novice). The designer could vary input parameters to generate predictions for other scenarios. The UI model could also include an equation similar to Equation 3.1 for predicting the number of errors.

$$\begin{aligned} T &= num\_tasks * (task\_time[task\_type] * \\ &\quad complexity\_adj[complexity\_type] * user\_adj[user\_type]) \end{aligned} \tag{3.1}$$

Equation 3.2 demonstrates a similar way to predict navigation time; this calculation could be embedded within a simple Web navigation model and used to generate predictions for the content organization approaches. This model is based on the taxonomy of Web tasks discussed in [Byrne *et al.* 1999]. Similarly to the previous example, the designer could specify input parameters for the number of pages (five in the first design and eight in the second), the complexity of pages (high in the first design and medium in the second), and the user type (novice). The UI model could also include an equation similar to Equation 3.2 for predicting the number of errors.

$$\begin{aligned} T &= num\_pages * ((navigate\_time * complexity\_adj[complexity\_type] * \\ &\quad user\_adj[user\_type]) + (read\_time * complexity\_adj[complexity\_type] * \\ &\quad user\_adj[user\_type]) + (think\_time * complexity\_adj[complexity\_type] * \\ &\quad user\_adj[user\_type])) \end{aligned} \tag{3.2}$$

Assuming models of graphical interface usage and Web site navigation existed, designers could quickly, albeit possibly not very accurately, compare designs and use predictions to inform decisions. Using empirical data to develop system parameters should improve the accuracy of such models.

3.10 New UE Simulation Methods

3.10.1 Simulation in Performance Evaluation

In simulation, the evaluator constructs a detailed model of a system to reproduce its behavior [Jain 1991]. Typically, a computer program (known as a simulator) exercises this underlying model with various input parameters and reports resulting system performance. Unlike an analytical model, which represents a high-level abstraction of system behavior, a simulator mimics system behavior. Hence, it is possible to use actual execution traces to drive a simulator, which is not possible with analytical models (see below). Consequently, analysts regard simulation results as more credible and accurate than analytical modeling results [Jain 1991]. Simulators also allow for the study of alternative designs before actually implementing the system. This is not possible with measurement techniques because the system must exist to capture performance data.

The underlying simulation model is one of the major differences among the various simulation approaches. The defining characteristics of these models are: the way the system evolves (time dependent or time independent), how its parameters are generated, and the number of states it can evolve into. In time-independent evolution, the system does not change characteristics based on time (i.e., time is not a system parameter); the opposite is true of time-dependent evolution. System parameters can be *fixed* (i.e., set to specific values) or *probabilistic* (i.e., randomly generated from probability distributions). Finally, simulation models can have a *finite* or countable number of system states or an *infinite* or uncountable number.

Another distinguishing feature of a simulator is its workload format. The workload may be in the form of fixed or probabilistic parameters that dictate the occurrence of various system events or an execution trace captured on a real system. Performance analysts consider trace-driven simulations to be the most credible and accurate [Jain 1991].

Table 3.8 summarizes characteristics for two frequently-used simulation approaches, discrete-event and Monte Carlo simulation. Discrete-event simulations model a system as it evolves over time by changing system state variables instantaneously in response to events. Analysts use this approach to simulate many aspects of computer systems, such as the processing subsystem, operating system, and various resource scheduling algorithms. Execution traces are often used in discrete-event simulations, since it is relatively easy to log system events.

Monte Carlo simulations model probabilistic phenomena that do not change characteristics with time. Analysts use this approach to conduct what-if analysis (i.e., predict resulting performance due to system resource changes) and to tune system parameters. Due to the random nature of Monte Carlo simulations, execution traces are not usually used with this approach. However, a Monte Carlo simulator may output an execution trace to facilitate analysis of its results.

3.10.2 Simulation in Usability Evaluation

Of the simulation methods surveyed in Chapter 2, all can be characterized as discrete-event simulations, except for information scent modeling [Chi *et al.* 2000]; information scent modeling is a close approximation to Monte Carlo simulation. Typical events modeled in these simulators include keystrokes, mouse clicks, hand and eye movements, as well as retrieving information from

Simulation Workload	Simulation Type	
	Discrete-event	**Monte Carlo**
Parameter Evolution	time-dependent	time-independent
Parameters	probabilistic	probabilistic
# of States	finite	finite
Trace Evolution	time-dependent	—
Parameters	probabilistic	—
# of States	finite	—

Table 3.8: Characteristics of simulation techniques in the performance evaluation domain.

Challenge	PE	UE
Modeling System	software program, simulation environment	UI development environment
Capturing Traces	record during measurement	usage traces (real & generated)
Using Traces	simulator reads & "executes"	simulate UI behavior realistically
Multiple Usage Scenarios	vary simulator parameters, multiple traces	usage traces (real & generated)

Table 3.9: Summary of the challenges in using PE simulation techniques in the UE domain. The PE column describes how these challenges are resolved within the PE domain, and the UE column describes potential ways to resolve these challenges within the UE domain.

memory. All of these simulation methods, except AMME (Automatic Mental Model Evaluator) [Rauterberg and Aeppili 1995], use fixed or probabilistic system parameters instead of usage traces; AMME constructs a petri net from usage traces.

3.10.3 Applying PE Simulation to UE

The underexplored simulation areas, discrete-event simulation with usage traces and Monte Carlo simulation, are promising research areas for automated UE. Several techniques exist for capturing traces or log files of interface usage [Neal and Simons 1983; Zettlemoyer *et al.* 1999]. As previously discussed, tools exist for automatically generating usage traces for WIMP [Kasik and George 1996] and Web [Chi *et al.* 2000] interfaces. One approach is to use real traces to drive a detailed UI simulation in the same manner discussed for measurement; AMME provides an example of this type of simulation. Such simulators would enable designers to perform what-if analysis and study alternative designs with realistic usage data.

Monte Carlo simulation could also contribute substantially to automated UE. Most simulations in the UE domain rely on a single user model, typically an expert user. One solution is to integrate the technique for automatically generating plausible usage traces into a Monte Carlo simulator; information scent modeling is a similar example of this type of simulation. Such a simulator could mimic uncertain behavior characteristic of novice users. This would enable designers to perform what-if analysis and study design alternatives with realistic usage data. Furthermore, the simulation run could be recorded for future use with a discrete-event simulator.

Table 3.9 summarizes challenges for using simulation in the UE domain as it is used in PE. The next section contains an in depth discussion of simulation solutions for the example tasks.

3.10.4 Example Simulators

For both of the example tasks, the assumption is that the UI is in the early design stages and consequently not available for running workloads. Hence, the designer must first construct a model to mimic UI behavior for each operation. The simplest way to accomplish this would be to expand a UI development environment (UIDE) or UI management system (UIMS) to support simulation. These environments enable a designer to specify a UI at a high-level (e.g., a model) and automatically generate an implementation.

Trace-driven discrete-event simulation is appropriate for simulating interface tasks such as the label creation example. However, all of the discrete-event simulators, except AMME, do not support executing usage traces. The drawback of AMME is that it requires log files captured during interface usage. The requirement of interface usage can be mitigated with a number of techniques previously discussed for automatically generating usage traces. Such traces could be saved in a format that a discrete-event simulator can process. In particular, traces need to be augmented with timing information.

The major difference between Monte Carlo and discrete-event simulation, especially simulation driven by usage traces, is that the task sequence is not pre-determined in a Monte Carlo simulation. This type of simulation is appropriate for the information-centric Web site example as described in the following section.

Monte Carlo Simulator for Information-Centric Web Sites

As previously mentioned, Monte Carlo simulation is appropriate for simulating information-centric Web sites, since this methodology does not require explicit task sequences. The survey in Chapter 2 revealed two related simulation methods: WebCriteria's Site Profile [Web Criteria 1999] and information scent modeling [Chi *et al.* 2000]. Site Profile attempts to mimic a user's information-seeking behavior within a model of an implemented site. It uses a idealist Web user model (called Max) that follows an explicit navigation path through the site, estimates page load and navigation times for the shortest path between the starting and ending points, and measures content freshness and page composition (amount of text and graphics). Currently, it does not use other user models, attempt to predict navigation paths, or consider the impact of other page features, such as the number of colors or fonts, in estimating navigation time. This simulation approach is more consistent with discrete-event simulation than Monte Carlo simulation, since it uses explicit navigation paths.

Information scent modeling is more consistent with Monte Carlo simulation and was developed for generating and capturing navigation paths for subsequent visualization. This approach creates a model of an existing site that embeds information about the similarity of content among pages, server log data, and linking structure. The evaluator specifies starting points in the site and information needs (target pages) as input to the simulator. The simulation models a number of agents (hypothetical users) traversing the links and content of the site model. At each page, the model considers information "scent" (i.e., common keywords between an agent's goal and content on linked pages) in making navigation decisions. Navigation decisions are controlled probabilistically such that most agents traverse higher-scent links (i.e., closest match to information goal) and some agents traverse lower-scent links. Simulated agents stop when they reach the target pages or after an arbitrary amount of effort (e.g., maximum number of links or browsing time).

The simulator records navigation paths and reports the proportion of agents that reached target pages. The authors comparison of actual and simulated navigation paths for Xerox's corporate site revealed a close match when scent is "clearly visible" (meaning links are not embedded in

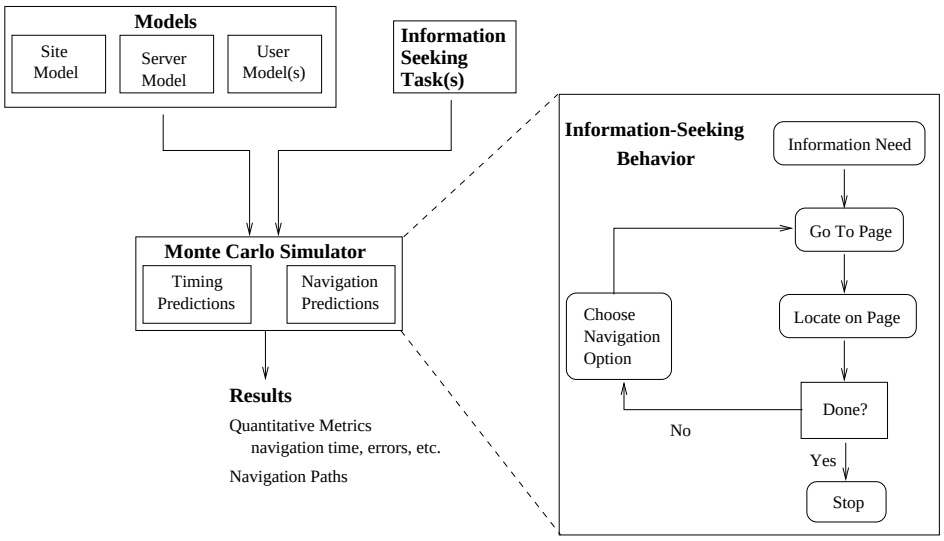


Figure 3.9: Proposed simulation architecture.

long text passages or obstructed by images). Since the site model does not consider actual page elements, the simulator cannot account for the impact of various page aspects, such as the amount of text or reading complexity, on navigation choices. Hence, this approach may enable only crude approximations of user behavior for sites with complex pages.

This section details a Monte Carlo simulation approach to address the limitations of the Site Profile and information scent modeling methodologies; some of this discussion was previously published as a poster [Ivory 2000]. Figure 3.9 depicts a proposed simulation architecture and underlying model of information-seeking behavior based on a study by Byrne *et al.* [1999].

A typical design scenario entails the designer initially creating several designs (i.e., site models) either by specifying information about each page, including the page title, metadata, page complexity and link structure, or importing this information from an existing site. The page complexity measure needs to consider various page features, such as the number of words, links, colors, fonts, reading complexity, etc.; the benchmarking approach discussed in Chapter 4 as well as the profiles developed in Chapter 6 provides insight for automatically determining a page complexity measure for existing sites, while estimates could be used for non-existent sites. The designer would also specify details about the server's latency and load (server model) and users' information tasks (e.g., destination pages and associated topics of interest). Finally, the designer would specify models of anticipated users with key parameters, such as the reading speed, connection speed, probability that a user will complete a task, read a page, make an error, etc. The designer could also specify constraints in the user model, such as an upper bound on navigation time or a small screen size, using production rules. It is possible to develop user models based on observed user behavior and reuse these models for simulation studies.

After specifying these models, the designer would then run the simulator for each design. Each run of the simulator would require the following steps. First, pick a starting page. There are three different models for how to do this: (1) User specified, (2) Randomly chosen independent of task (this may be used for assessing reachability), and (3) Chosen based on the task (this is equivalent to following a link returned by a search engine, a link from an external page, or a link from a usage trace). Next, repeat the steps below until either (a) the target page is reached; or (b) a stopping criteria is reached (e.g., maximum navigation time, all paths from starting point

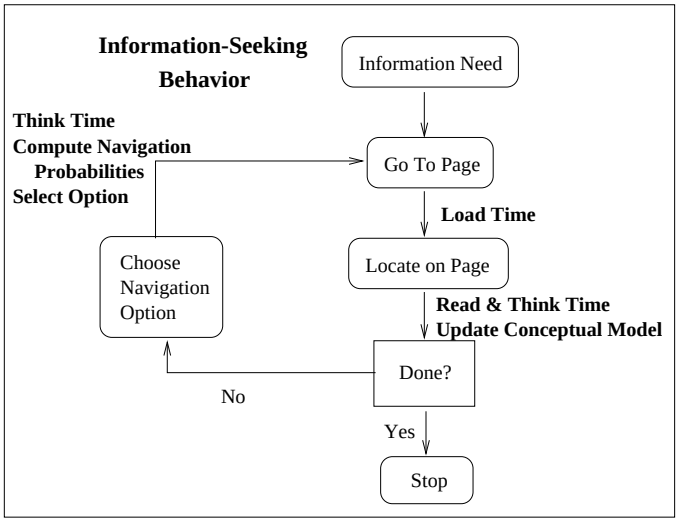


Figure 3.10: Simulator behavior during a run.

exhausted, or maximum number of links traversed). These steps are depicted in Figure 3.10.

Assuming the starting page is already loaded in the user's browser, the steps for simulating navigation are:

1. Read the page
   - This entails computing a read time based on the following:
     (a) page complexity (e.g., amount of text and reading difficulty)
     (b) page visitation (if previously viewed, then less read time)
   - Update the system clock after computing the read time.
2. Make a decision
   - Think time considers time for
     (a) deciding if target is reached (if so, end simulation run)
     (b) if not, deciding what to do next (this should be proportional to the number of options). This entails deciding on a navigation option (e.g., following a link or using the back button) as follows:
        i. survey options (create a list of choices)
        ii. compute a probability for selecting each option based on criteria described below; this procedure is not followed for the discrete-event mode, since the choice is dictated by the navigation trace.
        iii. prune options: if the probability for an option falls below a certain threshold, then eliminate the option.
        iv. choose option: if there is more than one option remaining, then the Monte Carlo algorithm determines the choice; this choice could be based on history (i.e., a user model or record of previous choices) or random.
   - Update the system clock after computing the think time.
3. Navigate

- This entails making the selected page the current page and marking it as visited. It also entails estimating a link traversal time and updating the system clock. This could be a fixed page download time or a variable that takes into account network variations, caching, and characteristics of the page itself.

Step 2b above requires computation of a probability for selecting a next move based on certain criteria. These include:

1. Metadata match: do a pairwise comparison of metadata between the current and potential page to compute a score indicating relatedness of content (it should be possible to pre-compute these for all links and store them in a structure); do a pairwise comparison of metadata between the target information and next page to compute another match probability; multiply the probabilities to compute a final match probability.

   Another possibility is to maintain a composite metadata analysis that's updated at each link. This composite analysis would take into account metadata on the previous pages traversed and the target information. This composite metadata could then be used in the pairwise comparison to compute a final match probability.

   Information foraging theory [Pirolli 1997; Pirolli *et al.* 1996] could be used for the metadata match algorithm. This approach has been used to present users with relevant Web pages in a site [Pirolli *et al.* 1996] and for information scent modeling [Chi *et al.* 2000].

2. Page visitation: if a page has been visited before, adjust selection probability according to the user model under use.

3. User model: adjust the match probability based on the simulated user. For example, if we are simulating a user learning by exploration, then the system state would reflect the user's current learning. If a page under consideration is consistent with a user's learning, then we would increase the match probability. Note that this criteria is different from (2), since it affects pages that have not been visited before.

The simulator could report simulated navigation time along with navigation traces to facilitate analysis and to possibly use with a discrete-event simulator. The designer could use simulation results for multiple site designs to determine the best information architecture for a site and to inform design improvements.

3.11 Summary

Using PE as a guiding framework provides much insight into creating new fully-automated (i.e., does not require interface use) UE methods. The analysis showed that the major challenges to address include: automatically generating high-level usage traces; mapping these high-level traces into UI operations; constructing UI models; and simulating UI behavior as realistically as possible. This chapter described solutions to these challenges for two example tasks.

The proposed simulation and analytical modeling approaches should be useful for helping designers choose among design alternatives before committing to expensive development costs. The proposed usability benchmarks should help assess implemented systems globally, across a wide range of tasks.

The remainder of this dissertation focuses on using PE as a guiding framework for developing a measurement approach for assessing Web interface quality. Specifically, it describes the

development of a Web interface benchmark consisting of over 150 page-level and site-level measures, the development of statistical models of highly-rated interfaces from these quantitative measures, and the application of these models in assessing Web interface quality and Web design guidelines.

Chapter 4

Automated Analysis Methodology and Tools

4.1 Introduction

This chapter presents an automated methodology for analyzing Web interfaces that leverages the benefits of both performance evaluation and usability evaluation as discussed in the previous chapters. In the spirit of measurement techniques in the performance evaluation domain, benchmarking in particular, the methodology entails computing an extensive set of quantitative page-level and site-level measures. These measures can be viewed as a benchmark because they enable objective comparison of Web interfaces. Specifically, these measures are used to derive statistical models of highly-rated interfaces. As is done with guideline review methods in the usability evaluation domain, the models are then used in the automated analysis of Web pages and sites. Unlike other guideline review methods, the guidelines are in essence derived from empirical data. Hence, it is also possible to use the models to validate or invalidate many guidelines proposed in the Web design literature.

This chapter and the subsequent two chapters detail the steps followed in this dissertation towards developing an automated analysis method for Web interfaces. First, this chapter presents two analysis scenarios as motivation for the methodology. Then, the methodology and tools are discussed. Chapter 5 summarizes an extensive survey of Web design literature culminating in the development of 157 page-level and site-level quantitative measures; these measures are computed by the Metrics Computation Tool discussed in this chapter. These measures are then used to develop statistical models of highly-rated Web interfaces in Chapter 6; these statistical models are incorporated into the Analysis Tool discussed in this chapter.

4.2 Analysis Scenarios

4.2.1 Web Interface Evaluation

As background for the methodology presented in this chapter, Figure 4.1 depicts an analysis scenario: a Web designer seeking to determine the quality of an interface design. If the site has already been designed and implemented, the designer could use the site as input to an analysis tool. The analysis tool (or benchmark program) would then sample pages within the site and generate a number of quantitative measures pertaining to all aspects of the interface. As discussed in Chapter 2, a key component of benchmarking is the ability to determine how well benchmark results com-

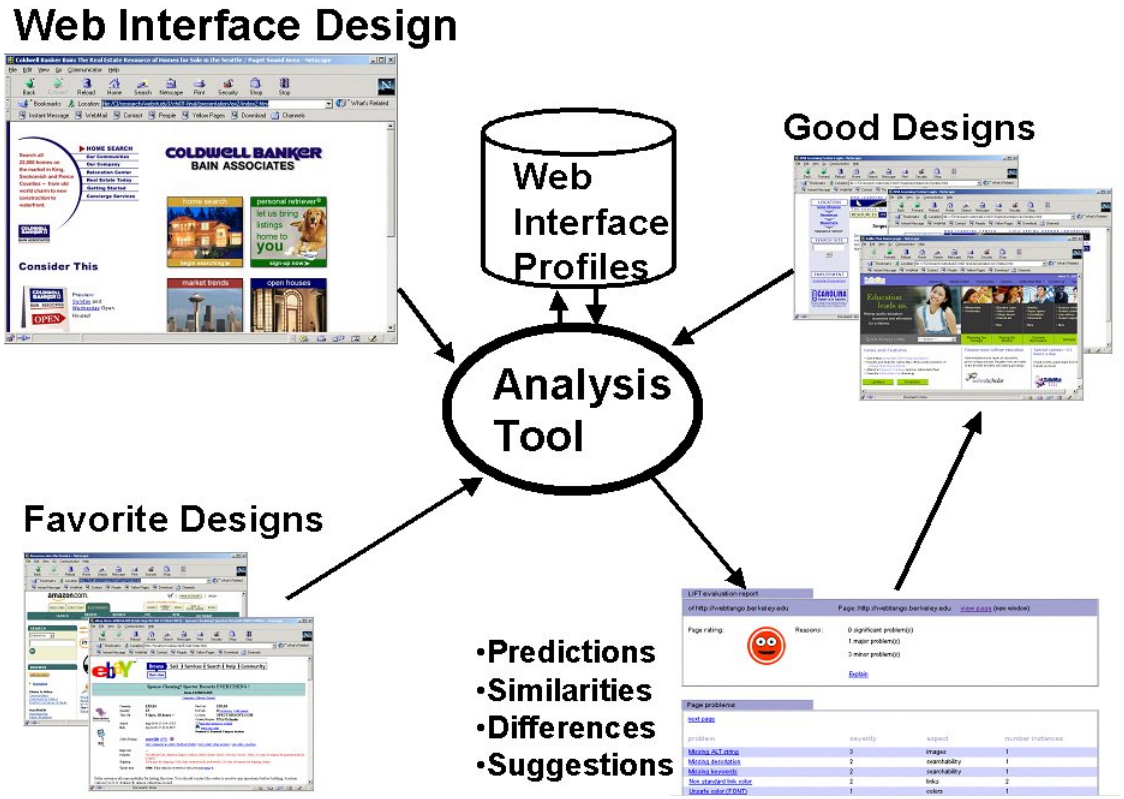


Figure 4.1: A Web interface evaluation scenario.

pare to other systems or a best case. For this scenario, designs that have been rated favorably by expert reviewers or users could be used for comparison purposes. Hence, the analysis tool could compare the input design's quantitative measures to those for highly-rated designs. Ideally, the analysis tool goes beyond traditional benchmarking and generates a report containing an interface quality or usability prediction, links to similar highly-rated designs from the comparison sample, differences and similarities to the highly-rated designs, and specific suggestions for improvements. The designer could use these results to choose between alternative designs as well as to inform design improvements. This analysis process could be iterated as necessary.

Similarly, the designer could use the analysis tool to explore results for other interface designs, such as favorite sites. This process may help to educate the designer on subtle aspects of design that may not be apparent from simply inspecting interfaces. The methodology and tools developed in this dissertation were designed to support many aspects of this Web interface evaluation scenario. Currently, recommending design improvements and presenting comparable designs are not supported; future work will focus on these aspects. However, all other aspects of the scenario are fully supported.

4.2.2 Web Interface Design

One can also imagine that a designer would want to obtain feedback on interface designs earlier during the design phase as opposed to after implementation. If the tool supported analysis of designs represented as images or templates, then it would be possible to support this evaluation. In particular, the tool needs to use image processing during analysis. One could also imagine supporting the designer even earlier in the design process, such as during the design exploration

and refinement phases [Newman and Landay 2000]. Given a large collection of favorably-rated sites, the designer could explore this collection to stimulate design ideas similarly to practices employed in the architecture domain [Elliott 2001]. During the design exploration phase, the designer could look for ways to organize content within health sites as well as navigation schemes, for example. During the design refinement phase, the designer may look for good page layouts, color palettes, site maps, navigation bars, form designs, etc.

Ideally, characteristics of Web pages and sites can be represented in a way to facilitate easily identifying pages and sites that satisfy queries similar to the ones above. Task-based search techniques that exploit metadata [Elliott 2001; English *et al.* 2001; Hearst 2000] should be helpful; Hearst [2000] proposes an approach wherein search interfaces present users with metadata facets for refining search results. Elliott [2001] presents a similar approach for exploring large online collections of architecture images; metadata describes the content of images, including location, architect, style, and kind of building. Metadata for Web interfaces could consist of the quantitative measures developed in this dissertation (discussed in Chapter 5) as well as others that describe for instance the size of the site, the type of site, page size, a page's functional type, elements on a page (e.g., navigation bars), as well as site ratings. Many of these measures could be computed automatically by the analysis methodology presented in this chapter. These measures could be organized into metadata facets. Scapin *et al.* [2000] presents a useful and relevant framework for organizing Web guidelines that includes a taxonomy of index keys (e.g., alignment, buttons, downloading, headings, language, scrolling, navigation structure, and so on); this taxonomy could be used to organize the measures into metadata facets.

4.3 Methodology

Figure 4.2 depicts the methodology developed to support the interface evaluation scenario. Earlier work on this methodology was published in [Ivory *et al.* 2000; Ivory *et al.* 2001]. The approach was developed mainly for information-centric Web interfaces (sites whose primary tasks entail locating specific information) as opposed to functionally-oriented interfaces (sites wherein users follow explicit task sequences). However, this approach could be used to some degree for functionally-oriented interfaces, since these interfaces typically present information as well.

The analysis methodology consists of two distinct but related phases: 1. establishing an interface quality baseline; and 2. analyzing interface quality. Both phases share common activities: crawling Web sites to download pages and associated elements (Site Crawling); and computing page-level and site-level quantitative measures (Metrics Computation). During the first phase, the page-level and site-level measures coupled with expert ratings of sites are analyzed to determine profiles of highly-rated interfaces. These profiles encapsulate key quantitative measures, thresholds for these measures, and effective relationships among key measures; thus, representing an interface quality baseline. During the second phase, the page-level and site-level measures are compared to the developed profiles and are used to assess interface quality. Sites analyzed in the latter phase are usually not the same as the sites used to develop profiles. Once profiles are developed, the analysis phase can be used on an ongoing basis. However, the interface quality baseline phase needs to be repeated periodically (annually or semi-annually) to ensure that profiles reflect current Web design practices.

This analysis methodology is consistent with other guideline review methods discussed in Chapter 2. It is also consistent with benchmarking methods discussed in Chapter 3. Specifically, it includes a synthetic benchmark that mimics a Web browser loading Web pages and uses an internal driver (i.e., one program for loading and running the workload). It also includes an automated

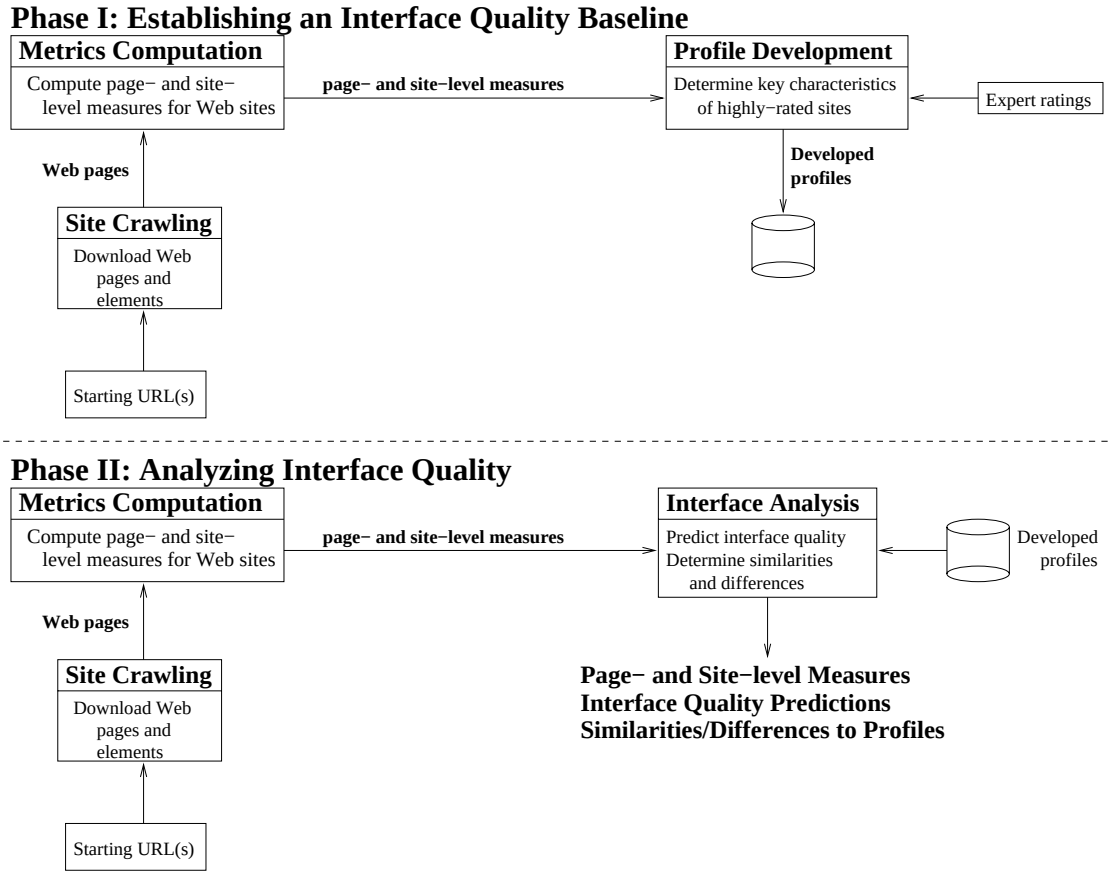


Figure 4.2: Overview of the analysis methodology.

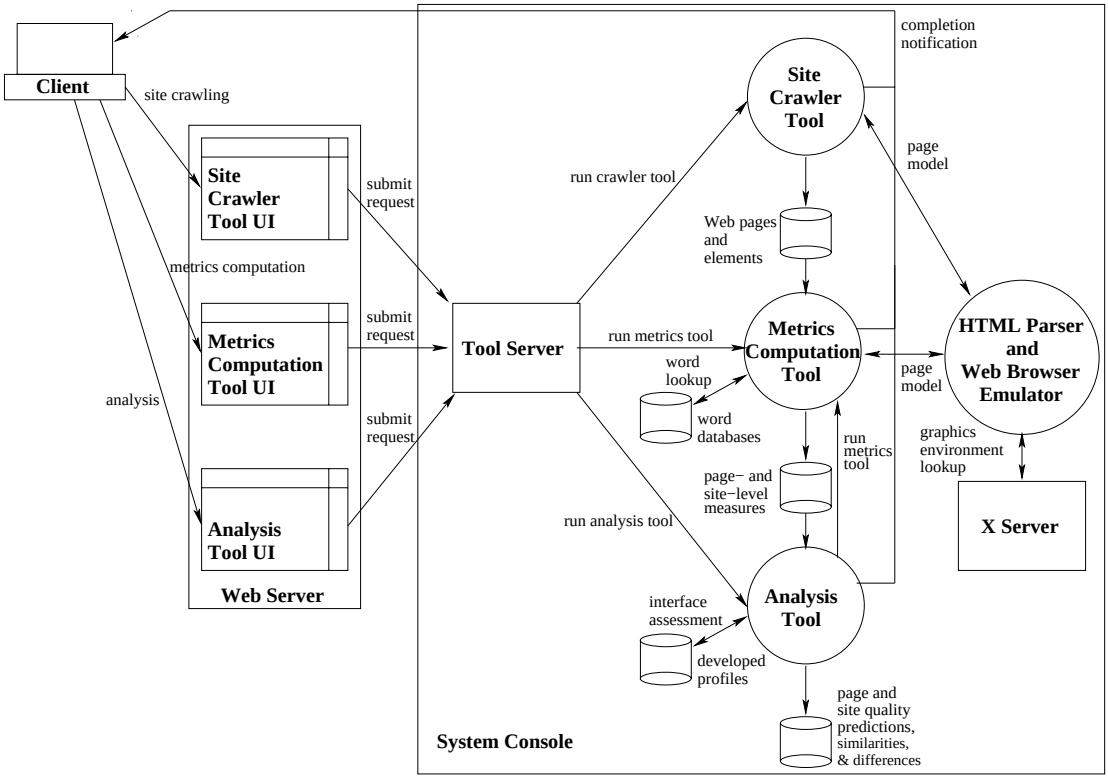


Figure 4.3: Architecture of tools developed to support the analysis methodology. All tools are available as part of the WebTango Research Project.

analysis component similar to operationalized guidelines. What distinguishes this analysis approach from other guideline review methods is: 1. the use of quantitative measures; 2. the use of empirical data to develop guidelines; and 3. the use of profiles (highly-rated interfaces as determined by expert ratings) as a comparison basis.

4.4 Tools

Figure 4.3 depicts the architecture of tools developed to support the analysis methodology. The following steps were taken to support the profile development discussed in Chapter 6.

1. Page were downloaded from numerous sites and stored on a Web server (site crawling).
2. Page-level and site-level measures were computed for each downloaded page and site (metrics computation).

Profiles developed in Chapter 6 are incorporated into the Analysis Tool. The Analysis Tool invokes the Metrics Computation Tool to generate measures; thus, eliminating the need to compute measures as a separate step.

This process is available to the public via separate interfaces for the site crawling and analysis steps; future work will integrate these interfaces into one UI to support the entire process. The interface for each tool routes requests to a server daemon (the Tool Server) for processing; the daemon in turn forks new processes to forward requests to the appropriate backend tool (Site

Crawler, Metrics Computation, or Analysis Tool). Both the Site Crawler and Metrics Computation Tools interact with the HTML Parser and Browser Emulator; this component creates a detailed representation of a Web page, including the x and y location of each page element, the width and height of each element, the font used, foreground and background color, and other attributes. The browser emulator determines many of the details, such as the height and width of text, by querying the graphics environment via the X Server running on the system console.

Currently, each tool sends an email notification to the client when the request completes; this notification includes a link to a tarred and gzipped file containing the output of running the tool. The output of running each tool is used as input to the subsequent step. For example, pages downloaded by the Site Crawler Tool are then processed by the Metrics Computation Tool to output page-level and site-level measures. Future work will focus on developing an interactive, integrated tool.

The components depicted in Figure 4.3 comprise over 33,000 lines of Java, HTML, and PERL code; they are discussed in detail in the remainder of this section. Appendix C provides information about running the tools.

4.4.1 HTML Parser and Browser Emulator

The Multivalent Browser (developed as part of the Berkeley Digital Library Project) [Phelps 1998; Phelps 2001] was used as a starting point for the HTML Parser and Browser Emulator depicted in Figure 4.3[1]. The Multivalent Browser enables creators of digital documents to represent their documents at multiple layers of abstraction and to annotate them with behaviors (e.g., actions), highlighting, and edit marks. The browser provides a means for these documents to be distributed and viewed by others. A variety of document formats are supported – OCR output, zip, ASCII, XML, and TeX – in addition to HTML. The browser parses HTML similarly to the Netscape Navigator 4.75 Browser's method and supports stylesheets. It does not support framesets, scripts, and other objects, such as applets.

The Multivalent Browser was revised extensively (~60% of code changed) to generate a more detailed page model for use by the Site Crawler and Metrics Computation Tools. Most of the revisions focused on enumerating frames, images, links, and objects that appear in a Web page and annotating each node in the page's tree structure with information about how the element is formatted (e.g., bolded, colored, italicized, etc.), whether it is a link (and if so whether it is an internal or external link), and downloading page elements to determine their sizes. The new parser also performs numerous corrections of HTML errors (e.g., out of order tags and tables without <tr> or <td> tags) while processing Web pages.

The new parser and browser emulator was configured to simulate the Netscape Navigator 4.75 browser with default settings (fonts, link colors, menu bar, address bar, toolbar, and status line). Currently, most monitors are 800 x 600 pixels [DreamInk 2000; Nielsen 2000]; thus, the window size was fixed at 800 x 600 pixels. The current implementation does not support pages that use framesets, although it does support pages with inline frames; given that Netscape 4.75 does not support inline frames, alternative HTML is typically provided in Web pages by designers and can be processed. Future work may entail using the Mozilla or Opera parser and browser to support more accurate page rendering, framesets, and scripts as well as to address performance problems with the current tool.

The HTML Parser and Browser Emulator comprises approximately 20,000 lines of Java code. The tool requires an average of 90 seconds to generate a model of a Web page. Performance should be substantially improved by using a more robust parser and browser.

[1] The Multivalent Browser code was provided courtesy of Tom Phelps.

4.4.2 Site Crawler Tool

A special Web site crawler was developed to address limitations of existing crawlers, such as Wget [GNU Software Foundation 1999] and HTTrack [Roche 2001]. Existing crawlers rely on path information to determine the depth of pages, which can be somewhat misleading. They are also not selective in determining pages to download; they download advertisements and Macromedia Flash pages, for instance. Finally, they typically attempt to mirror the directory structure of the remote site or place all images, stylesheets, etc. into one directory; this makes it challenging to relocate an individual page and its associated elements.

Like existing crawlers, the Site Crawler Tool is multi-threaded. In addition, the tool has the following key features.

- Pages at multiple levels of the site are accessed, where level zero is the home page, level one refers to pages one link away from the home page, level two refers to pages one link away from the level one sites, and so on. The standard settings are: download the home page, up to 15 level-one pages and 45 level-two pages (3 from each of the level-one pages).
- Links are selected for crawling such that: they are not advertisements, guestbooks, Flash pages, login pages, chatrooms, documents, or shopping carts; and they are internal to the site.
- Each downloaded page is stored in a directory with all images, stylesheets, frames, and objects (e.g., scripts and applets) aggregated and stored in subdirectories. This makes it easy to relocate pages and associated elements.

The Site Crawler Tool replaces links to images, stylesheets, and other page elements on the remote server with the corresponding locally-stored files. It also creates input files for the Metrics Computation Tool.

The Site Crawler Tool comprises approximately 1,500 lines of Java, PERL, and HTML code; the Web interface for submitting crawling requests is implemented in HTML and PERL, while the actual crawler is implemented in Java. The tool spends at most twelve minutes downloading pages on a site before aborting.

4.4.3 Metrics Computation Tool

The Metrics Computation Tool is consistent with other benchmarks discussed in Chapter 3: it is portable to other platforms due to its implementation in Java; it produces quantitative measures for comparison; and its results are reproducible (i.e., successive runs of the tool on the same page produce the exact same results). The tool computes 141 page-level and 16 site-level measures; these measures assess many facets of Web interfaces as discussed in Chapter 5. Numerous heuristics were developed for computing these measures, such as detecting headings, good color usage, internal links, and graphical ads. The tool also uses a number of auxiliary databases, such as the MRC psycholinguistic database [Coltheart 2001] for determining spelling errors and the number of syllables in words, and other lists of acronyms, abbreviations, medical terms, and common (stop) words. Currently, the tool only processes English text.

The Metrics Computation Tool comprises approximately 10,500 lines of Java, PERL, and HTML code. The Web interface for submitting requests for metrics computation is implemented in HTML and PERL. The page-level measures are implemented in Java, while the between-page and site-level measures are implemented in Java and PERL. The tool requires on average two minutes to compute page-level measures per page, 30 seconds to compute between-page measures per page,

and one minute to compute site-level measures per site. The performance bottleneck with page-level measures is the need to traverse the page model at least three times to annotate nodes and compute measures. During the first phase, the HTML Parser and Browser Emulator constructs an initial page model. Then, the Metrics Computation Tool traverses the page model to annotate headings, sentences, links, and to store link text. Page-level measures are computed during the final pass through the revised page model.

4.4.4 Analysis Tool

The Analysis Tool encompasses several statistical models for assessing Web page and site quality. Statistical models include decision trees, discriminant classification functions, and K-means cluster models as discussed in Chapter 6. For cluster and discriminant classification models, the top ten measures that are similar and different from highly-rated interfaces are reported; acceptable measure values are also provided. The cluster models also report the distance between measure values on a page and measure values at the cluster centroid; the distance reflects the total standard deviation units of difference across all measures. Currently, the Analysis Tool does not provide the designer with example good sites or pages; future work will focus on developing an algorithm to provide examples. Future work will also focus on supporting automated critique or providing explicit suggestions for improvements as well as interactive analysis.

The Analysis Tool comprises about 1,500 lines of PERL and HTML code. Each of the profiles discussed in Chapter 6 is implemented in PERL, while the interface for submitting analysis requests is implemented in HTML and PERL. The tool requires one minute on average to compute page and site quality predictions for a site.

4.5 Summary

This chapter presented an analysis methodology consistent with measurement approaches used in the performance evaluation domain and guideline review approaches used in the usability evaluation domain. Unlike other Web assessment techniques, this approach uses quantitative measures and empirical data to develop guidelines for comparison purposes.

Although the tools are very robust, they suffer from several limitations. Currently, the crawling and analysis tools are separate processes and could benefit from being consolidated. Another limitation is that metrics computation is extremely compute intensive. Timings on a Sun Enterprise 450 server revealed that it requires about four minutes on average to process each Web page to compute page-level measures, between-page measures, and page quality assessments. An additional minute is needed to compute site-level measures and the quality assessments. It could take an hour to analyze the quality of a site, including all of the steps from crawling 10 pages on the site to generating site quality assessments; this depends largely on the complexity of pages in terms of page sizes and the number of associated elements, including images and stylesheets. Hence, extensive optimization is needed to enable this process to occur in real time. One possibility is to reimplement the crawling, metrics computation, and analysis processes using a robust, open source browser, such as Mozilla or Opera. This will also enable the tools to support framesets and script processing, since robust browsers support these elements.

The major limitation of this approach is that the quantitative measures do not capture users' subjective preferences. For example, one study has shown that perceived download speed is more important than actual download speed [Scanlon and Schroeder 2000a]. Although it is possible to measure actual download speed, it may not be possible to assess perceived speed. Nonetheless,

the methodology can be viewed as a reverse engineering of design decisions that were presumably informed by user input.

Chapter 5

Web Interface Measures

5.1 Introduction

There is an abundance of design recommendations, recipes, and guidelines for building usable Web sites [Flanders and Willis 1998; Fleming 1998; Nielsen 1998c; Nielsen 1999b; Nielsen 2000; Rosenfeld and Morville 1998; Sano 1996; Schriver 1997; Shedroff 1999; Shneiderman 1997; Spool *et al.* 1999]. These guidelines address a broad range of Web site and page features, from the amount of content on a page to the breadth and depth of pages in the site. However, there is little consistency and overlap among them [Ratner *et al.* 1996] making it difficult to know which guidelines to adhere to. Furthermore, there is a wide gap between a heuristic such as "make the interface consistent" and the operationalization of this advice. Finally, most recommendations have not been empirically validated.

This chapter presents a set of 157 page-level and site-level measures based on an extensive survey of design recommendations from recognized experts and usability studies. The intent is to first quantify features discussed in the literature and to then determine their importance in producing highly-rated designs. Statistical models developed in Chapter 6 should facilitate the development of concrete, quantitative guidelines for improving Web interfaces; Chapter 10 demonstrates this for a subset of design guidelines.

This chapter begins with a view of Web interface structure and a summary of the 157 measures. The summary is followed by detailed discussions of all quantitative measures. Appendix C provides instructions for accessing an interactive appendix with visual depictions of all of the measures. Chapter 6 explores the use of these measures in developing profiles of quality Web interfaces.

5.2 Web Interface Structure

A Web interface is a mix of many elements (text, links, and graphics), formatting of these elements, and other aspects that affect the overall interface quality. Web interface design entails a complex set of activities for addressing these diverse aspects. To gain insight into Web design practices, Newman and Landay [2000] conducted an ethnographic study wherein they observed and interviewed eleven professional Web designers. One important finding was that most designers viewed Web interface design as being comprised of three components – information design, navigation design, and graphic design – as depicted in the Venn diagram in Figure 5.1. Information design focuses on determining an information structure (i.e., identifying and grouping content items) and developing category labels to reflect the information structure. Navigation design fo-

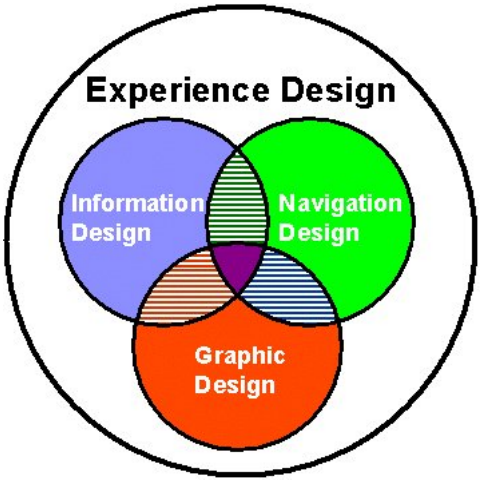


Figure 5.1: Overview of Web interface design. The Venn diagram is a modified version of the one in [Newman and Landay 2000]; it is reprinted with permission of the authors.

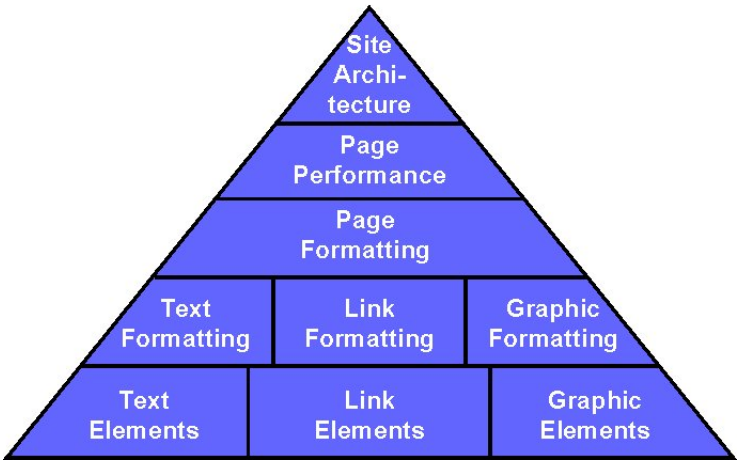


Figure 5.2: Aspects associated with Web interface structure.

cuses on developing navigation mechanisms (e.g., navigation bars and links) to facilitate interaction with the information structure. Finally, graphic design focuses on visual presentation and layout. All of these design components affect the overall quality of the Web interface. The Web design literature also discusses a larger, overarching aspect – experience design [Creative Good 1999; Shedroff 2001] – the outer circle of Figure 5.1. Experience design encompasses information, navigation, and graphic design. However, it also encompasses other aspects that affect the user experience, such as download time, the presence of graphical ads, popup windows, etc.

Information, navigation, graphic, and experience design can be further refined into the aspects depicted in Figure 5.2. The figure shows that text, link, and graphic elements are the building blocks of Web interfaces; all other aspects are based on these. The next level of Figure 5.2 addresses formatting of these building blocks, while the subsequent level addresses page-level formatting. The top two levels address the performance of pages and the architecture of sites, including the consistency, breadth, and depth of pages. The bottom three levels of Figure 5.2 are associated with information, navigation, and graphic design activities, while the top two levels – Page Performance and Site Architecture – are associated with experience design.

Metric	Description
Word Count	Total words on a page
Body Text %	Percentage of words that are body vs. display (i.e., headings) text
Emphasized Body Text %	Portion of body text that is emphasized (e.g., bold, capitalized or near !'s)
Text Positioning Count	Changes in text position from flush left
Text Cluster Count	Text areas highlighted with color, bordered regions, rules, or lists
Link Count	Total links on a page
Page Size	Total bytes for the page and images
Graphic %	Percentage of page bytes for images
Graphics Count	Total images on a page
Color Count	Total colors used
Font Count	Total font face, size, bolding, and italics combinations
Reading Complexity	Gunning Fog Index (ratios of words, sentences and words with more than 3 syllables)

Table 5.1: Web page metrics used in prior studies [Ivory *et al.* 2000; Ivory *et al.* 2001]. All measures, except Reading Complexity, were examined in both studies; Reading Complexity was not included in the second study.

Aspects presented in Figures 5.1 and 5.2 are used to organize discussions throughout this chapter.

5.3 Summary of Web Interface Measures

An extensive survey of Web design literature, including texts written by recognized experts (e.g., [Fleming 1998; Nielsen 2000; Sano 1996; Spool *et al.* 1999]) and published user studies (e.g., [Bernard and Mills 2000; Bernard *et al.* 2001; Boyarski *et al.* 1998; Larson and Czerwinski 1998]) was conducted to identify key features that impact the quality and usability of Web interfaces [Ivory *et al.* 2000]. HTML style guides were not consulted because they have been shown to be highly inconsistent [Ratner *et al.* 1996]. Sixty two features were identified from the literature, including: the amount of text on a page, fonts, colors, consistency of page layout in the site, use of frames, and others. As part of the analysis methodology, 157 quantitative measures were then developed to assess many of the 62 features.

Previous work by the author showed that twelve Web interface measures – Word Count, Body Text Percentage, Emphasized Body Text Percentage, Text Positioning Count, Text Cluster Count, Link Count, Page Size, Graphic Percentage, Graphics Count, Color Count, Font Count, and Reading Complexity – could be used to accurately distinguish pages from highly-rated sites [Ivory *et al.* 2000; Ivory *et al.* 2001]. Table 5.1 describes these measures. For this dissertation, 157 quantitative measures (including nine of the original twelve measures and variations of the other three) were developed to further assess aspects of the information, navigation, graphic, and experience design of Web interfaces. These measures provide some support for assessing 56 of the 62 features (90%) identified as impacting usability in the Web design literature. Measures developed in previous studies assessed less than 50% of these 62 features.

Guidelines provided in Section 3.4.1 for determining performance metrics were considered in developing the 157 measures. Specifically, a subset of measures with the following characteristics

were implemented.

- **Low Variability:** measures are not a ratio of two or more variables; there is one exception to this rule – the average number of words in link text – which was developed to assess a feature reported in the literature.
- **Nonredundancy:** two measures do not convey essentially the same information.
- **Completeness:** measures reflect all aspects of the Web interface (i.e., information, navigation, graphic, and experience design).

A sample of fourteen Web pages with widely differing characteristics was used to validate the implemented measures. The actual value of each measure was manually computed and then used to determine the accuracy of computed results. For each page and each measure, the number of accurate hits and misses as well as the number of false positives and negatives were determined as described below.

Accurate Positive: an element is counted as it should have been counted (e.g., an actual word is counted by the tool as a word).

Accurate Negative: an element is not counted and it should not have been counted (e.g., an actual display word is not counted as a body word). This is relevant for discriminating measures that can label an object more than one way, such as deciding whether a word is a heading, body, or text link word.

False Positive: an element is counted and it should not have been counted (e.g., an actual body word is counted as a display word).

False Negative: an element is not counted and it should have been counted (e.g., an actual body word is not counted as a body word).

False positives and false negatives typically occur for discriminating measures (i.e., ones that entail deciding between two or more options). After counting the four types of hits and misses for a measure on a page, the following accuracies were then computed for the measure.

Hit Accuracy: the ratio of the number of accurate positives to the sum of the number of accurate positives and false negatives. False positives are not included in this computation, since they represent inaccurate hits.

Miss Accuracy: the ratio of the number of accurate negatives to the sum of the number of accurate negatives and false positives. False negatives are not included in this computation, since they represent inaccurate misses.

The average hit and miss accuracies for a measure is then the average hit and miss accuracies across all sample pages. Finally, the overall accuracy is computed over the average hit and miss accuracies. Metric tables in this chapter report the average hit, average miss, and overall accuracy for each measure across the 14-page sample. With a few exceptions, all of the measures are highly accurate (>84% overall accuracy across the sample). The least accurate measures – text positioning count (number of changes in text alignment from flush left) and text and link text cluster counts (areas highlighted with color, rules, lists, etc.) – require image processing as discussed later.

For measures developed in this chapter, the HTML Parser and Browser Emulator (see Chapter 4) was configured to simulate the Netscape Navigator 4.75 browser with default settings (fonts, link colors, menu bar, address bar, toolbar, and status line). The window size was fixed at 800 x 600 pixels.

The remainder of this chapter presents the 62 features derived from the literature survey, the measures developed to assess these features, and a short discussion of specific guidance from the literature when available. The discussion reflects the hierarchy presented in Figure 5.2. In each section, summary tables depict the measures, the Web interface aspects (information, navigation, graphic, and experience design) assessed by the measures, and their accuracy as described above. Two classes of measures were developed: discriminating (deciding between two or more options) and non-discriminating (a count). For discriminating measures, hit and miss accuracies are reported; they are not reported for non-discriminating measures. The overall accuracy is reported for both discriminating and non-discriminating measures.

5.4 Text Element Measures

Tables 5.2, 5.3, and 5.4 summarize 31 text element measures derived from the literature survey and discussed in this section. These measures provide insight about the following Web page features.

1. How much text is on the page?
2. What kind of text is on the page?
3. How good is the text on the page? Good in this context refers to whether or not the text contains an abundance of common words typically referred to as stop words. Good words are not stop words.
4. How complex is the text on the page? Complexity refers to the reading level required to understand the text as determined by the Gunning Fog Index [Gunning 1973].

5.4.1 Text Element Measures: Page Text

The text or visible (legible) words on a Web page has been discussed extensively in the literature [Flanders and Willis 1998; Landesman and Schroeder 2000; Nielsen 2000; Schriver 1997; Stein 1997] and is considered a major component of the information design. An analysis of experts' ratings of Web sites submitted for the Webby Awards 2000 revealed that content was by far the best predictor of ratings [Sinha *et al.* 2001]. The literature includes the following heuristics.

- Users prefer pages with more content as opposed to breaking content over multiple pages [Landesman and Schroeder 2000].
- Keep text short; use 50% less text than in print publications [Nielsen 2000].
- Break text up into smaller units on multiple pages [Flanders and Willis 1998; Nielsen 2000].

As is often found in the literature, the first guideline contradicts the other two. Furthermore, there is no concrete guidance on how much text is enough or too much. Thus, a Word Count measure was developed to assess the amount of text on the page. The presence of invisible

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How much text is on the page?								
Word Count	Total visible words	√				–	–	99.8%
Page Title Word Count	Number of words in the page's title (max of 64 chars)	√	√			–	–	100.0%
Overall Page Title Word Count	Number of words in the page's title (no char max)	√	√			–	–	100.0%
Invisible Word Count	Number of invisible words				√	–	–	100.0%
Meta Tag Word Count	Number of words in meta tags				√	–	–	100.0%
What kind of text is on the page?								
Body Word Count	Words that are body text (i.e., not headings or links)	√				99.9%	99.5%	99.8%
Display Word Count	Words that are display text (i.e., headings that are not links)	√				97.5%	100.0%	98.7%
Display Link Word Count	Words that are both link text and headings	√	√			75.5%	100.0%	87.7%
Link Word Count	Words that are link text and are not headings	√	√			99.7%	99.5%	99.6%
Average Link Words	Average number of words in link text		√			–	–	100.0%
Graphic Word Count	Number of words from <img> alt attributes	√				–	–	100.0%
Ad Word Count	Number of words possibly indicating ads ('advertisement' or 'sponsor')	√				100%	100%	100%
Exclamation Point Count	Number of exclamation points	√				–	–	100.0%
Spelling Error Count	Number of misspelled words				√	100.0%	99.9%	100.0%

Table 5.2: Summary of text element measures (Table 1 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How good is the text on the page?								
Good Word Count	Total good visible words (i.e., not stop words)	√				100.0%	100.0%	100.0%
Good Body Word Count	Good body text words	√				100.0%	100.0%	100.0%
Good Display Word Count	Good display text words	√				100.0%	100.0%	100.0%
Good Display Link Word Count	Good combined display and link text words (i.e., not stop words or 'click')	√	√			100.0%	100.0%	100.0%
Good Link Word Count	Good link text words	√	√			100.0%	99.7%	99.8%
Average Good Link Words	Average number of good link text words		√			100.0%	100.0%	100.0%
Good Graphic Word Count	Number of good words from <img> alt attributes	√				100.0%	100.0%	100.0%
Good Page Title Word Count	Number of good page title words (max of 64 chars)	√	√			100.0%	100.0%	100.0%
Overall Good Page Title Word Count	Total number of good page title words (no char max)	√	√			100.0%	100.0%	100.0%
Good Meta Tag Word Count	Number of good meta tag words				√	100.0%	100.0%	100.0%

Table 5.3: Summary of text element measures (Table 2 of 3). Good in this context refers to the use of words that are not stop words or the word 'click' in the case of text links. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How complex is the text on the page?								
Reading Complexity	Gunning Fog Index computed over prose	√				–	–	97.6%
Overall Reading Complexity	Gunning Fog Index computed over all text	√				–	–	97.9%
Fog Word Count	Number of prose words (for Reading Complexity)	√				99.5%	100.0%	99.7%
Fog Big Word Count	Number of big prose words (for Reading Complexity)	√				94.9%	97.6%	96.2%
Overall Fog Big Word Count	Number of big words (for Overall Reading Complexity)	√				96.5%	98.5%	97.5%
Fog Sentence Count	Number of sentences (for Reading Complexity)	√				–	–	98.6%
Overall Fog Sentence Count	Number of sentences (for Overall Reading Complexity)	√				–	–	98.6%

Table 5.4: Summary of text element measures (Table 3 of 3). Complexity in this context refers to the reading level required to understand the text; this is determined by the Gunning Fog Index [Gunning 1973]. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

words (i.e., words formatted with the same foreground and background colors, thus making them illegible) is also measured, since such words may indicate spamming (techniques used to influence search engine ranking). These counts only include words in the static HTML and do not include words embedded in images, applets, and other objects. Image processing techniques are required to count words in these cases.

Word Count was examined in both of the prior metric studies [Ivory *et al.* 2000; Ivory *et al.* 2001] and was shown to be a key measure for distinguishing highly-rated pages in the second study. Furthermore, dividing the sample into three groups – low word count (average of 66 words), medium word count (average of 230 words), and high word count (average of 820 words) – revealed several key differences between highly-rated and poorly-rated Web pages.

5.4.2 Text Element Measures: Page Title

A page's title is also an important element of the information design [Berners-Lee 1995; Flanders and Willis 1998; Nielsen 2000]. Specific guidance includes the following.

- Use no more than 64 characters [Berners-Lee 1995].
- Use 2–6 words (40–60 characters) [Flanders and Willis 1998].
- Use different page titles for each page [Nielsen 2000].

To assess the number of words used in page titles, two measures – Page Title Word Count and Overall Page Title Word Count – were developed. The difference between these two measures is a restriction of 64 characters on the Page Title Word Count but not on the Overall Page Title Word Count. Prior experience with the Page Title Word Count revealed that extremely long page titles (e.g., >200 words) were sometimes used, possibly as a spamming technique. However, long titles are also used legitimately to make search engine results more readable; further analysis of good and poor Web pages needs to be conducted to know for certain that spamming is used. Nonetheless, only a portion of these words are actually visible in a Web browser; thus, the guideline from the HTML 2.0 specification (64 maximum characters) [Berners-Lee 1995] is used to determine the number of visible page title words. Deviations between these two measures may indicate spamming. To assess whether page titles varied between pages within a site, scent quality and page title consistency measures (discussed in Sections 5.12.7 and 5.13.1) were implemented.

5.4.3 Text Element Measures: Page Abstract

Although a page's abstract (i.e., meta tags used for search engines) is not a visible page feature, it does affect the experience design, especially when searching is used to locate information. Nielsen [2000] suggests that Web designers use meta tags with less than 150–200 characters on the page. To assess the use of meta tags, a Meta Tag Word Count was developed. It includes words used in both the description and keyword attributes of the meta tag.

5.4.4 Text Element Measures: Body Text

The interplay of display (i.e., headings) and body text affects users ability to scan the information on a page [Nielsen 1997; Schriver 1997]. Ideally, meaningful headings and sub-headings are used to help users locate specific information [Nielsen 1997]. Although no specific guidance was provided with respect to the amount of body vs. display text in a Web page, a Body Word Count

and several Display Word Count measures were developed to assess the balance between these two features.

Detecting headings within a Web page is not a straight-forward activity. Prior experience revealed that only a small fraction of Web pages actually use the HTML header tags (<h1>, <h2>, <h3>, etc.) to format headings; the majority of pages use font and stylesheet tags. Hence, several heuristics were developed to detect headings; heuristics are based on the sample of fourteen Web pages used throughout metrics development. For example, if text is bolded, not embedded in other non-bolded text, and is not in a text link, then it is considered to be a heading. Another heuristic examines whether text is emphasized in some other manner (e.g., italicized or colored), is not embedded in other non-emphasized text, is formatted with a font size greater than twelve points, and is not in a text link to determine whether the text is a heading. The fixed font size of 12 points is not ideal, because it would be more appropriate to use font sizes used in the document as a baseline. The heuristics detected headings in the Web page sample with 98.7% overall. Similar heuristics were developed for detecting link headings with 87.7% overall accuracy; the hit accuracy was only 75.5%, while the miss accuracy was 100%. Using image processing techniques on an image of a Web page may result in more accurate detection of link headings.

The two prior metric studies used a Body Text Percentage measure (ratio of body text words to all words on the page). Both studies showed this measure to be important for distinguishing highly-rated pages. Despite its importance, this measure was abandoned because it directly violates the low variability (not a ratio of two or more measures) guidance for performance metrics. Analysis of the Word and Body Word Counts can provide the same insight as the Body Text Percentage measure.

5.4.5 Text Element Measures: Link Text

Words used in text links affect both the information and navigation design. Although the total number of link text words is measured by the Rating Game tool [Stein 1997], no specific guidance was found in the literature with respect to the appropriate amount of link text on a page. Two link text measures were developed – Link Word Count and Display Link Word Count. The latter measure reports the number of link text words that are also headings.

5.4.6 Text Element Measures: Link Text Length

The previous section discussed the total number of link text words, but this section focuses on the number of words in a text link. The literature states that the number of words used in each text link affects the information design and the navigation design in particular [Chi *et al.* 2000; Nielsen 2000; Sawyer and Schroeder 2000; Spool *et al.* 1999]. Specific guidance includes the following.

- Use 2–4 words in text links [Nielsen 2000].
- Use links with 7–12 "useful" words [Sawyer and Schroeder 2000].

These two guidelines are obviously contradictory. Hence, an Average Link Words measure was developed to assess the length of link text. This measure is the ratio of the total number of text link words (i.e., Link and Display Link Word Counts) to the number of text links. This ratio violates the low variability guidance for performance metrics; however, the measure was implemented because it allows for direct assessment of a feature with contradictory guidance in the literature. The second guideline includes a qualifier – "useful" words – this is assessed to some

degree with a complementary measure, Average Good Link Words, which is discussed in Section 5.4.12.

5.4.7 Text Element Measures: Content Percentage

Nielsen [2000] suggests that the portion of the Web page devoted to content should represent 50–80% of the page. The content percentage can be computed by highlighting the portion of the page devoted to content and then determining what percentage of the page it represents. It is difficult to compute this percentage as prescribed without using image processing techniques. Instead, the following indirect measures were developed: the total number of words (Section 5.4.1); and the total number of words used for text links (Section 5.4.5).

5.4.8 Text Element Measures: Navigation Percentage

Similarly to the content percentage, Nielsen [2000] suggests that the portion of the Web page devoted to navigation should represent 20% of the page; this percentage should be higher for home and intermediate pages. The navigation percentage can be computed by highlighting the portion of the page devoted to navigation and then determining what percentage of the page it represents. It is difficult to compute this percentage as prescribed without using image processing techniques. Several indirect measures were developed, including: the total number of links of various kinds (e.g., text and graphic; Section 5.5.1); and the total number of words used for text links (Section 5.4.5).

5.4.9 Text Element Measures: Exclamation Points

The use of exclamation points in the page text has been discussed in the literature [Flanders and Willis 1998; Stein 1997]. The consensus is to minimize the use of exclamation points, since they are equivalent to blinking text [Flanders and Willis 1998]. The literature does not provide guidance on an acceptable number of exclamation points. Hence, an Exclamation Point Count was developed; this measure includes exclamation points in body, display, and link text.

5.4.10 Text Element Measures: Typographical Errors

As part of the Web Credibility Project at Stanford University [Kim and Fogg 1999; Fogg *et al.* 2000], a survey of over 1,400 Web users was conducted to identify aspects that impact the credibility of Web interfaces. Survey results showed that an amateurism factor, which includes the presence of spelling and other typographical errors, was negatively correlated with credibility [Fogg *et al.* 2000]. A follow-up controlled study further established that the presence of typographical errors decreased credibility [Kim and Fogg 1999]. The obvious guidance from this research is to avoid typographical errors. Flanders and Willis [1998] also suggest that Web designers avoid making typographical mistakes on pages.

The Metrics Computation Tool assesses one type of typographical error – spelling errors (English only). An extensive dictionary of English words, computer, Internet, medical terms, acronyms, and abbreviations, including the MRC psycholinguistic database [Coltheart 2001], is used for checking spelling errors. One limitation of the spelling error measure is that it ignores capitalized words, since they may be proper nouns typically not found in dictionaries. Hence, the number of spelling errors may be underreported. Another limitation is that it may count jargon words as spelling errors; thus, inflating the number of spelling errors. Other typographical

errors, such as misplaced punctuation, are not assessed with quantitative measures, since more sophisticated computational linguistics is required.

5.4.11 Text Element Measures: Readability

The literature survey revealed numerous discussions of the readability or required reading level of text [Berners-Lee 1995; Flanders and Willis 1998; Gunning 1973; Nielsen 2000; Schriver 1997; Spool *et al.* 1999]. Spool *et al.* [1999] determined that the Gunning Fog Index (GFI) [Gunning 1973] was the only readability measure correlated with Web interfaces. To compute this index, a passage containing at least 100 words needs to be selected from the page. Then, the number of words (Fog Word Count), the number of words with more than two syllables (Fog Big Word Count), and the number of sentences (Fog Sentence Count) needs to be computed. Equation 5.1 demonstrates how to compute the Gunning Fog Index from these measures.

$$GFI = \left(\frac{Fog\ Word\ Count}{Fog\ Sentence\ Count} + \frac{Fog\ Big\ Word\ Count}{Fog\ Word\ Count} * 100.0 \right) * 0.4 \qquad (5.1)$$

This measure was originally developed for printed documents; however, Spool *et al.* [1999] discovered that this measure correlated with the scannability of Web pages. Specific guidance for the GFI in both the print and Web domains is provided below. As can be expected, guidance is contradictory in the two domains.

- A lower Gunning Fog Index of 7–8 is ideal for printed documents [Gunning 1973]; only a 7th or 8th grade education is required to read them.
- A higher Gunning Fog Index (~15.3) is ideal for Web pages; pages that are reported to require a college education to read them facilitate scanning [Spool *et al.* 1999].

Two reading complexity measures were developed – the Reading Complexity computed over prose text (sentences) and the Overall Reading Complexity computed over all of the text, including links and bulleted lists. Prior experience with computing the Gunning Fog Index over all of the text suggested that this measure was not truly a measure of reading complexity, rather it measured the degree to which text is broken up to facilitate page scanning [Ivory *et al.* 2000; Spool *et al.* 1999]. Hence, the reading complexity measure computed over just the prose text was developed to be more consistent with the intended use of the Gunning Fog Index.

All of the measures used in computing the reading complexity measures, such as the number of sentences and big words, are reported by the Metrics Computation Tool. The number of big words is determined by first looking up words in the MRC psycholinguistic database [Coltheart 2001], which contains the number of syllables for over 100,000 words. If the word is not in the database, then the algorithm used by the UNIX style program (counting consonant vowel pairs) [Cherry and Vesterman 1981] is used.

A previous study of the reading complexity computed over all of the page text showed this measure to be important for distinguishing unrated (from sites that have not been identified by reputable sources as exhibiting high quality) and rated (from sites that have been identified by reputable sources as exhibiting high quality) pages [Ivory *et al.* 2000]; this study is discussed briefly in Chapter 6. The reading complexity of rated pages was consistent with the second guideline presented above. The study also showed that reading complexity values above 15.8 were not associated with rated pages; thus, there appears to be a threshold above which higher reading complexity (i.e., reported as more difficult to read) may impede scanning.

5.4.12 Text Element Measures: Information Quality

The literature extensively discusses ways to improve the quality of information (e.g., content appropriateness, relevance, language, tone, and freshness) [Flanders and Willis 1998; Nielsen 2000; Rosenfeld and Morville 1998; Schriver 1997; Spool *et al.* 1999]. Specific guidance includes the following.

- Support efficient, easy first-time use (i.e., logical grouping of content and organization) [Rosenfeld and Morville 1998].
- Update content often [Flanders and Willis 1998; Nielsen 2000].

It is extremely difficult to assess the quality of information without user input. Hence, the implemented measures provide only limited coverage of the broad spectrum of aspects that influence information quality. Specifically, the Metrics Computation Tool computes the number of good words, link words, display words, body words, page title words, meta tag words, and so on. A word is determined to be good if it is not a stop word; an extensive list of 525 common English words is used for this assessment. For determining good link words, the word 'click' is also considered as a stop word. None of the quantitative measures assess the guidelines depicted above.

5.5 Link Element Measures

Table 5.5 summarizes six link element measures derived from the literature survey and discussed in this section. (These measures are different than the link text measures presented in the previous section.) These measures provide insight about the following Web page features.

1. How many links are on the page?
2. What kind of links are on the page?

5.5.1 Link Element Measures: Links

Links are an essential element of the navigation design and are discussed extensively in the literature [Flanders and Willis 1998; Furnas 1997; Larson and Czerwinski 1998; Rosenfeld and Morville 1998; Sano 1996; Schriver 1997; Spool *et al.* 1999; Zaphiris and Mtei 1997]. Several usability studies have been conducted to provide the following guidance about the breadth (i.e., how many links are presented on a page), depth (i.e., how many levels must be traversed to find information), and other aspects of the navigation structure.

- Use moderate levels of breadth with minimal depth (e.g., two levels) in the information architecture [Larson and Czerwinski 1998].
- Minimize depth [Zaphiris and Mtei 1997].
- A large number of links impedes navigation [Spool *et al.* 1999].
- Avoid broken links [Flanders and Willis 1998; Nielsen 2000; Spool *et al.* 1999].

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How many links are on the page?								
Link Count	Total number of links		√			–	–	99.8%
What kind of links are on the page?								
Text Link Count	Number of text links	√	√			99.7%	100.0%	99.9%
Link Graphic Count	Number of image links		√	√		100.0%	100.0%	100.0%
Page Link Count	Number of links to other sections (i.e., anchors) within the page		√			100.0%	100.0%	100.0%
Internal Link Count	Number of links that point to destination pages within the site		√			100.0%	100.0%	100.0%
Redundant Link Count	Number of links that point to the same destination page as other links on the page		√			100.0%	100.0%	100.0%

Table 5.5: Summary of link element measures. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

A Link Count was developed to assess the total number of links (graphical and text) on a page. This measure captures the breadth of links on a page, but it does not assess the depth of links. It was examined in both of the prior metric studies and found to be important for distinguishing highly-rated pages. The maximum page depth measure (discussed in Section 5.13.2) provides some insight into navigation structure. For instance, the deepest level that the crawler could traverse on the site (i.e., not find pages that hadn't been seen before) may reflect the depth of the information architecture.

Broken links have been mentioned repeatedly in the literature as a usability problem. Given that there are numerous tools for detecting broken links, such as Weblint [Bowers 1996], no measure was developed.

5.5.2 Link Element Measures: Text Links

Text links are considered to be the most important type of links in the literature [Flanders and Willis 1998; Sawyer and Schroeder 2000; Scanlon and Schroeder 2000c; Spool *et al.* 1999]. The consensus in the literature is that text rather than image links should be used [Flanders and Willis 1998; Spool *et al.* 1999]. A Text Link Count was developed to measure the number of text links on a Web page.

5.5.3 Link Element Measures: Link Graphics

The literature extensively discusses the use of graphical links [Flanders and Willis 1998; Sano 1996; Sawyer and Schroeder 2000; Scanlon and Schroeder 2000c; Spool *et al.* 1999]. Specific guidance includes the following.

- Avoid using graphical text links; they are typically ignored [Sawyer and Schroeder 2000; Spool *et al.* 1999] or may impede navigation [Scanlon and Schroeder 2000c; Spool *et al.* 1999].
- Use corresponding text links [Flanders and Willis 1998; Sano 1996].

A Link Graphic Count was developed to assess the number of links that are images. A similar measure was computed for the number of graphics that are also links and is discussed in Section 5.6.2. Although the Link Graphic Count measures the use of graphical links, there is no direct measure of whether equivalent text links is provided. However, the Redundant Link Count (discussed in Section 5.5.8) may provide some indirect insight about the use of equivalent text links.

5.5.4 Link Element Measures: Within-Page Links

The use of within-page links or links that point to other areas in the same page has been found to be problematic [Nielsen 2000; Sawyer and Schroeder 2000; Spool *et al.* 1999]. The consensus is that these types of links should be avoided, since they may be confusing [Nielsen 2000; Sawyer and Schroeder 2000; Spool *et al.* 1999]. A Page Link Count was developed to report the presence of within-page links.

5.5.5 Link Element Measures: External Links

The use of external links or links that point to other sites has also been reported as potentially problematic [Nielsen 2000; Spool *et al.* 1999]. One rationale is that users may not be aware that they left the original site [Spool *et al.* 1999]. Nielsen [1997] presents the contrary view that external links increase credibility based on a study of nineteen users reading Web pages.

Nielsen [2000] suggests that Web designers use a different link color scheme to signify external links and to better inform users that they are leaving the site.

An Internal Link Count was developed to measure the use of internal as opposed to external links. This measure considers links that point to the same domain (e.g., www1.cnet.com and www2.cnet.com) as being internal. The difference between the Link and Internal Link Counts lies in the number of external links. No measure was developed to determine if different color schemes are used for internal and external links.

5.5.6 Link Element Measures: Embedded Links

Embedding links within text on the page has been discussed in the literature [Rosenfeld and Morville 1998; Spool *et al.* 1999]. The consensus is that Web designers should avoid surrounding links with text, since they are difficult to scan [Rosenfeld and Morville 1998; Spool *et al.* 1999]. No measure was developed to detect the presence of embedded links, since image processing is required for accurate detection.

5.5.7 Link Element Measures: Wrapped Links

Spool *et al.* [1999] suggests that wrapped links or links spanning multiple lines should be avoided. During usability studies, users interpreted each line of a wrapped link as being a separate link versus all lines comprising a single link. No measure was developed to detect the presence of wrapped links, since it requires an accurate simulation of browser rendering. Use of a browser that can more accurately represent the layout of page elements, such as Mozilla or Opera, would facilitate detection of wrapped links.

5.5.8 Link Element Measures: Redundant Links

The surveyed literature espouses the value of redundant or multiple links to the same content [Sawyer and Schroeder 2000; Spool *et al.* 1999; Spool *et al.* 2000], while one study of different types of links (e.g., neighborhood, parent, index, etc.) found redundant links to confuse users within cyber shopping malls [Kim and Yoo 2000]. Specific guidance includes the following.

- Use multiple links to the same content with appropriate scent in each area [Spool *et al.* 2000].
- Use different forms for repeated links (e.g., text, graphical text, or image) [Sawyer and Schroeder 2000].
- Redundant links may cause confusion [Kim and Yoo 2000].

A Redundant Link Count was developed to assess the use of repeated links. No measure was implemented to detect whether different forms of links are used or whether different link text is used to label links that point to the same location.

5.5.9 Link Element Measures: Navigation Quality

Much has been said about many aspects of the navigation structure, such as the clarity of links, the use of scent (hints about the contents on a linked page), the relevance of links, and the use of effective navigation schemes [Chi *et al.* 2000; Fleming 1998; Furnas 1997; Larson and Czerwinski 1998; Miller and Remington 2000; Nielsen 2000; Rosenfeld and Morville 1998; Sawyer *et al.* 2000; Spool *et al.* 1999; Spool *et al.* 2000]. Specific guidance on these aspects includes the following.

- Use clear headings with related links (i.e., link clustering) to enhance scent [Spool *et al.* 2000].
- Expose multiple levels of the information architecture (i.e., link clustering with headings) [Sawyer *et al.* 2000].
- Effective navigation requires small pages (views), few clicks between pages, and strong scent [Furnas 1997].
- Weak scent (i.e., ambiguous link text) impedes navigation [Chi *et al.* 2000; Miller and Remington 2000; Spool *et al.* 1999; Spool *et al.* 2000].
- Similar link text across links impedes navigation [Spool *et al.* 1999].
- Do not use a shell strategy (i.e., fixed navigation bar content); use navigation bars at the top and bottom of pages vs. down the sides [Spool *et al.* 1999].
- Avoid using 'Click Here' for link text [Nielsen 2000].
- Support multiple modes of finding information (i.e., directed searching and browsing) [Fleming 1998; Rosenfeld and Morville 1998].
- The navigation scheme should be easy to learn, consistent, efficient, and relevant to the type of site (i.e., entertainment and information sites use different navigation schemes) [Fleming 1998].
- Use breadcrumbs (i.e., displaying a navigation trail) vs. long navigation bars [Nielsen 2000].

Many of these suggestions are difficult to measure in an automated manner. However, some guidelines, such as the number of good link words (excludes the word 'click'; Section 5.4.12), support for searching (Section 5.10.7), use of link clustering (Section 5.7.7), and the consistency of link elements and formatting (Section 5.13.1), are measured. Several site architecture measures, such as the maximum page depth and breadth (as determined by the crawling depth and breadth; Section 5.13.2), may also provide some insight about the navigation structure.

5.6 Graphic Element Measures

Table 5.6 summarizes six graphic element measures developed and discussed in this section. These measures assess the following features of Web pages.

1. How many graphics are on the page?
2. What kind of graphics are on the page?

5.6.1 Graphic Element Measures: Graphics

Images are a key element of the graphic design, and image use is discussed extensively in the literature [Ambühler and Lindenmeyer 1999; Flanders and Willis 1998; Nielsen 2000; Sano 1996; Schriver 1997; Spool *et al.* 1999; Stein 1997]. Scanlon and Schroeder [2000c] identified and provided guidance for the following four categories of graphics.

- **Content Graphics** - provide content (i.e., see vs. read); users typically do not complain about the download speed of content graphics.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How many graphics are on the page?								
Graphic Count	Total number of images			√		–	–	100.0%
What kind of graphics are on the page?								
Redundant Graphic Count	Number of images that point to the same image files as other images			√		100.0%	100.0%	100.0%
Graphic Link Count	Number of images that are links		√	√		100.0%	100.0%	100.0%
Animated Graphic Count	Number of animated images			√		100.0%	100.0%	100.0%
Graphic Ad Count	Number of images that possibly indicate ads			√		97.9%	96.1%	97.0%
Animated Graphic Ad Count	Number of animated images that possibly indicate ads			√		97.2%	99.9%	98.6%

Table 5.6: Summary of graphic element measures. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

- **Navigation Graphics** - help users navigate; these should typically be avoided.
- **Organizer Graphics** - bullets, rules, etc. that direct users' attention on the page; these are typically ignored by users.
- **Ornamental Graphics** - logos and other images that ornament the page; these images are typically the least effective and most costly in terms of download speed.

In some cases, one graphic may serve multiple roles. Furthermore, the role of a graphic may not be apparent from looking at it. Regardless of the role of graphics, the consensus in the literature is that the number of images needs to be minimized to improve download speed. Nielsen [2000] also suggests that text rendered as images (one type of content graphic) should be eliminated except for image captions. A Graphic Count measure was developed to quantify the use of images on a Web page. Text rendered as images is not assessed, since it would require image processing. Another limitation of this measure is that it includes images that may not be visible on the page, such as spacer images; image processing techniques are required to accurately report the number of visible images on the page. The use of navigation graphics is assessed with the Graphic Links measure discussed in the following section. Although the use of organizer and ornamental graphics is not directly measured, the developed Redundant Graphic Count (repeated use of image files) may provide an indirect measurement of these types of images.

The Graphic Count measure was examined in both prior metric studies. The first study determined that rated pages contained more images than pages that were not rated; inspection of a random sample of pages revealed that this higher number of graphics was attributable to organizer graphics. The second study found that in all cases, highly-rated pages contained fewer images. (Results from the second study are more definitive because they are based on analysis of expert ratings rather than rated and unrated sites). Even though rated pages contained more images in the first study, graphic count was not a key predictor of rated or highly-rated pages in either study.

5.6.2 Graphic Element Measures: Graphical Links

The use of images as links is also discussed in the literature [Flanders and Willis 1998; Sawyer and Schroeder 2000; Scanlon and Schroeder 2000c; Spool *et al.* 1999]. Specific guidance about graphical links is below. This guidance was also reported for the Link Graphic Count discussed in Section 5.5.3. The Graphic Link Count was developed to quantify the use of images as links. This measure varies from the Link Graphic Count in Section 5.5.3 when image maps are used on pages, since every area of an image map is counted as a separate image link.

- Avoid using graphical text links; they are typically ignored [Sawyer and Schroeder 2000; Spool *et al.* 1999] or may impede navigation [Scanlon and Schroeder 2000c; Spool *et al.* 1999].
- Use corresponding text links [Flanders and Willis 1998; Sano 1996].

5.6.3 Graphic Element Measures: Graphical Ads

Several literature sources discuss the presence of graphical ads on Web pages [Kim and Fogg 1999; Klee and Schroeder 2000; Nielsen 2000]. Specific guidance includes the following.

- Ads affect the user experience; integrate ads with content [Klee and Schroeder 2000].
- Usability dictates that ads should be eliminated [Nielsen 2000].

- Ads increase credibility [Kim and Fogg 1999].

The guidelines are obviously contradictory. Kim and Fogg [1999] describe credibility as a "high level of perceived trustworthiness and expertise," which appears to be related to usability, although this link has yet to be established. A controlled study wherein 38 users rated Web pages (with and without graphical ads) on credibility showed that pages with graphical ads were rated as more credible than those without graphical ads. A Graphic Ad Count was developed to gain more insight about the use of ads on Web pages. This measure uses a number of heuristics (e.g., whether the image is a link to a known advertising agency or contains words indicative of advertisement in the URL) to detect the presence of ads with 97% overall accuracy.

5.6.4 Graphic Element Measures: Animation

The use of animated images and scrolling text is often debated in the literature [Flanders and Willis 1998; Nielsen 2000; Spool *et al.* 1999]. Specific guidance on animation includes the following.

- Minimize animated graphics [Flanders and Willis 1998].
- Avoid using animation unless it is appropriate (e.g., showing transitions over time) [Nielsen 2000].
- Animation is irritating to users; it impedes scanning [Spool *et al.* 1999].

None of these guidelines provide concrete guidance about how much animation is too much. Hence, an Animated Graphic Count was developed to quantify the use of animated images. Similarly, an Animated Graphic Ad Count was developed to quantify the use of animated graphical ads. Animated images are counted as ads if they are used as links to pages from well-known advertising agencies (DoubleClick, HitBox, Adforce, etc.), contain the word ad or advertising in the URL, or are to pages on external sites. No measure was developed to detect the use of scrolling text, since scrolling text is typically implemented using scripts.

5.6.5 Graphic Formatting Measures: Graphic Quality

The quality of images used on the page, including their appropriateness and optimization (i.e., bytes and resolution), is discussed in the literature [Flanders and Willis 1998; Nielsen 2000; Sano 1996; Schriver 1997; Spool *et al.* 1999]. The only concrete guidance provided is for Web designers to use smaller images with fewer colors [Flanders and Willis 1998; Sano 1996]. Several measures were developed to assess the sizes of images as well as the screen area covered by images (Section 5.9). Image processing is not used; thus, the use of colors within images is not assessed.

5.7 Text Formatting Measures

Tables 5.7, 5.8, and 5.9 summarize 24 text formatting measures developed and discussed in this section. These measures assess the following aspects of Web interfaces.

1. How is body text emphasized?
2. Is there underlined text that is not in text links?

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How is body text emphasized?								
Emphasized Body Word Count	Body text words that are emphasized (e.g., bolded or capitalized)	√		√		98.6%	98.1%	98.4%
Bolded Body Word Count	Body text words that are bolded	√		√		100.0%	99.9%	99.9%
Capitalized Body Word Count	Body text words that are capitalized	√		√		99.4%	99.7%	99.6%
Colored Body Word Count	Body text words that are a color other than the default text color	√		√		96.8%	95.1%	96.0%
Exclaimed Body Word Count	Body text words that are near exclamation points	√				100.0%	100.0%	100.0%
Italicized Body Word Count	Body text words that are italicized	√		√		100.0%	99.5%	99.8%
Is there underlined text (not in text links)?								
Underlined Word Count	Number of words that are underlined but are not text links	√	√	√		100.0%	100.0%	100.0%

Table 5.7: Summary of text formatting measures (Table 1 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

3. What font styles are used?
4. What font sizes are used?
5. How many text colors are used?
6. How many times is text re-positioned?
7. How are text areas highlighted?

5.7.1 Text Formatting Measures: Text Emphasis

Several literature sources provide guidance about emphasized text (e.g., bolded, italicized, and capitalized text) on the page [Flanders and Willis 1998; Nielsen 2000; Schriver 1997]. Specific guidance includes the following.

- Avoid mixing text attributes (e.g., color, bolding, and size) [Flanders and Willis 1998].

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
What font styles are used?								
Serif Word Count	Number of words formatted with serif font faces	√		√		99.9%	100.0%	99.9%
Sans Serif Word Count	Number of words formatted with sans serif font faces	√		√		99.9%	91.7%	95.8%
Undetermined Font Style Word Count	Number of words formatted with undetermined font faces	√		√		100.0%	100.0%	100.0%
Font Style	Whether text is predominately sans serif, serif, or undetermined font styles	√		√		–	–	100.0%
What font sizes are used?								
Minimum Font Size	Smallest font size (in points) used for text	√		√		–	–	100.0%
Maximum Font Size	Largest font size used for text	√		√		–	–	100.0%
Average Font Size	Predominate font size used for text	√		√		–	–	100.0%
How many text colors are used?								
Body Color Count	Number of colors used for body text	√		√		100.0%	95.6%	97.8%
Display Color Count	Number of colors used for display text	√		√		100.0%	100.0%	100.0%

Table 5.8: Summary of text formatting measures (Table 2 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How many times is text re-positioned?								
Text Positioning Count	Number of text areas that change position from flush left			√		–	–	87.7%
Text Column Count	Number of x positions (i.e., columns) where text starts			√		–	–	95.0%
How are text areas highlighted?								
Text Cluster Count	Number of text areas that are highlighted in some manner	√		√		–	–	68.3%
Link Text Cluster Count	Number of link text areas that are highlighted in some manner	√	√	√		–	–	77.8%
Border Cluster Count	Number of text and link text areas that are highlighted with bordered regions	√		√		77.8%	100.0%	88.9%
Color Cluster Count	Number of text and link text areas that are highlighted with colored regions	√		√		93.6%	84.7%	89.2%
List Cluster Count	Number of text and link text areas that are highlighted with lists	√		√		91.7%	100.0%	95.8%
Rule Cluster Count	Number of text and link text areas that are highlighted with horizontal or vertical rules	√		√		60.8%	98.8%	79.8%

Table 5.9: Summary of text formatting measures (Table 3 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

- Minimize blinking text [Flanders and Willis 1998; Nielsen 2000].
- Avoid italicizing and underlining text [Schriver 1997].
- Avoid using all caps for text [Nielsen 2000].

No measure was developed to assess text emphasis across the entire page, since emphasis is used for both display and body text. However, measures were developed to assess body text emphasis as discussed in the next section.

5.7.2 Text Formatting Measures: Body Text Emphasis

The guidance provided in the previous section also applies to emphasized body text; thus, it is repeated below.

- Avoid mixing text attributes (e.g., color, bolding, and size) [Flanders and Willis 1998].
- Minimize blinking text [Flanders and Willis 1998; Nielsen 2000].
- Avoid italicizing and underlining text [Schriver 1997].
- Avoid using all caps for text [Nielsen 2000].

Numerous measures were developed to quantify body text emphasis, including an Emphasized Body Word Count (total number of body words that are bolded, italicized, or emphasized in some other way) and Bolded, Italicized, and Colored Body Word Counts. An Underlined Word Count was also developed to detect the presence of words that are underlined but not in text links, since it is possible for such words to be mistaken for links. The metrics do not directly measure the mixing of text emphasis, although this may be apparent from the individual measures.

An Emphasized Body Text Percentage measure was examined during both prior metric studies. This measure did not make a significant contribution for distinguishing rated pages in the first study, but it did make a significant contribution for distinguishing highly-rated pages in the second one. This measure was abandoned because it violates the low variability guidance for performance metrics.

5.7.3 Text Formatting Measures: Font Styles

Font style (e.g., serif or sans serif) has been discussed extensively in the literature [Bernard and Mills 2000; Bernard *et al.* 2001; Boyarski *et al.* 1998; Nielsen 2000; Schriver 1997]. Several usability studies have been conducted wherein serif and sans serif fonts were compared with respect to reading speed and user preference [Bernard and Mills 2000; Bernard *et al.* 2001]. Boyarski *et al.* [1998] have also compared fonts designed for computer screens to those designed for print. Specific guidance on font styles includes the following.

- Use serif fonts for faster reading by older adults [Bernard *et al.* 2001].
- Sans serif fonts have a slight advantage over serif fonts and are more preferred [Bernard and Mills 2000; Schriver 1997].
- Use fonts designed for computer screens (e.g., Verdana and Georgia) rather than fonts designed for print (e.g., Times New Roman) [Boyarski *et al.* 1998].

- Use only a few sizes from one or two typeface families; use one serif and one sans serif font for contrast [Schriver 1997].
- Use sans serif fonts for smaller text and serif fonts for larger text [Nielsen 2000].

Measures were developed to determine the predominant font style used for text. Specifically, the number of words formatted with serif, sans serif, and undetermined font styles are computed. Tables of serif and sans serif font names were compiled from the literature and Web sites that classify fonts (e.g., www.buyfonts.com). The HTML Parser and Browser Emulator assumes that the first valid font name specified is the one used by the browser. The font style of a word is then determined by looking up the font name in the tables. A Font Style measure is also computed as the maximum over the font style word counts; it identifies the predominate font style (sans serif, serif, or undetermined font styles).

The measures do not capture whether fonts designed for computer screens as opposed to those designed for print are used, whether a few font sizes from multiple typeface families are used, or whether sans serif fonts are used for small text and serif fonts are used for larger text. However, these measure in conjunction with the font point size measures (discussed below) provide some indirect insight about the font face and size combinations used.

5.7.4 Text Formatting Measures: Font Point Sizes

Font sizes (e.g., 9 pt and 14 pt) used in Web pages has been discussed extensively in the literature [Bernard *et al.* 2001; Flanders and Willis 1998; Nielsen 2000; Schriver 1997]. Specific guidance on font sizes includes the following.

- Use 14 pt fonts for older adults [Bernard *et al.* 2001].
- Use font sizes greater than 9 pt [Flanders and Willis 1998; Schriver 1997].
- Use 10 to 11 pt (or higher) for body text and 14 pt (or higher) for display text; use larger point sizes for serif faces [Schriver 1997].

Measures were developed to assess the maximum, minimum, and average point sizes of text on Web pages. The average point size was determined by tracking point sizes used for each word on the page and then dividing by the total number of words (i.e., Word Count). No measures were developed to track fonts sizes used for body as opposed to display text or for sans serif as opposed to serif fonts.

5.7.5 Text Formatting Measures: Text Colors

Flanders and Willis [1998] encourages Web designers to minimize the number of text colors. In addition to the Colored Body Word Count (discussed in Section 5.7.2), Body and Display Text Color Counts were developed; these measures report the number of unique colors used for body and display text. The measures do not assess if different colors are used for body and display text.

One limitation of the color measures is that they may count colors that are not discriminable or perceptually close. For example, it is possible that users may consider text colored with two different shades of blue as being the same color; color counts would be inflated in this case. Future work will consider the perceptual closeness of colors in metrics computation.

5.7.6 Text Formatting Measures: Text Positioning

Changing the alignment of text has been discussed in the literature [Flanders and Willis 1998; Nielsen 2000; Schriver 1997], and specific guidance includes the following.

- Avoid centering large blocks of text or links; left-justified text is preferred [Flanders and Willis 1998].
- Use left-justified, ragged-right margins for text; do not center items in vertical lists [Schriver 1997].

A Text Positioning Count was developed to quantify the number of times that text areas change position from flush left. This measure was examined in both prior metric studies and determined to be important for distinguishing rated and highly-rated pages. A similar measure, Text Column Count, was developed to determine the number of unique positions or columns where text starts on the page. Ideally, this measure will provide some insight about the complexity of the page layout, since tables are typically used in a nested manner to control text positioning beyond using the alignment tags. No measure was developed to assess the alignment of items in a list or the amount of text that is aligned.

5.7.7 Text Formatting Measures: Text Clustering

Use of text clusters or highlighting text areas in some manner (e.g., enclosing text in bordered or colored regions) is encouraged as a way to emphasize important text on the page [Landesman and Schroeder 2000; Nielsen 2000; Sawyer and Schroeder 2000; Schriver 1997]. Text clustering is not to be confused with text emphasis; in the latter case, only the text is emphasized (e.g., bolded or italicized) as opposed to the entire area surrounding the text in the first case. Specific guidance on text clustering includes the following.

- Use text clustering in small amounts and clusters should not contain much continuous text [Schriver 1997].
- Delineate links with bullets, spaces, etc. when they are in a group [Landesman and Schroeder 2000; Sawyer and Schroeder 2000]. This type of formatting is considered to be a form of text clustering.

Several measures of text and link text clustering were developed, including Text and Link Text Cluster Counts. If more than 50% of the text in a cluster is link text, then the cluster is considered a link text cluster and vice versa for text clusters. Use of specific clustering techniques, such as colored, list, or rule clusters, is also quantified. For colored text clusters, proximity to other colored clusters is considered. For example, if a colored cluster is adjacent to another colored cluster with a perceptually-close color, then only one text cluster is counted. Perceptual color closeness is determined by first translating the RGB (red, green, and blue) colors [Foley *et al.* 1990] into CIE Luv (luminancy and chrominancy) colors [Bourgin 1994]. The CIE Luv color space is device-independent and uniform (i.e., all colors are equidistant from each other). If the Euclidean distance of two Luv colors is less than 5, then the colors are considered to be perceptually close [Jackson *et al.* 1994].

Detecting some forms of text clustering, such as manual rule or list clusters (see Sections 5.7.8 and 5.7.9 below), is not as accurate as all of the other measures developed; image processing techniques are required to improve the accuracy of these measures.

The Text Cluster Count (encompassing text and link text clusters) was studied in both prior metric studies. Although it was shown in both cases that rated and highly-rated pages used text clustering more so than the other pages, this measure did not play an important role in classifying these pages.

5.7.8 Text Formatting Measures: Lists

A list is one form of text clustering as discussed above. Schriver [1997] suggests that Web designers minimize the number of lists on a Web page. A List Cluster Count was developed to quantify the number of lists on the page. One limitation of this measure is that it can only capture lists created with HTML list tags (e.g., <ul> or <ol>). Hence, it was only 79% accurate overall with only a 60% hit accuracy for the Web page sample. Image processing techniques are required to detect manually-created lists (e.g., rows of items separated with
 tags) and to consequently improve accuracy.

5.7.9 Text Formatting Measures: Rules

Horizontal and vertical rules are other forms of text clustering. Specific guidance on the use of rules includes the following.

- Minimize the number of rules [Schriver 1997].
- Horizontal rules that are the full width of the page may be interpreted as the end of a page and possibly discourage scrolling [Spool *et al.* 1999].

A Rule Cluster Count was developed to quantify the number of rules on the page. One limitation of this measure is that it can only capture rules created with the HTML rule tag (<hr>). Image processing techniques are required to detect manually-created rules (e.g., an image containing a thin line). No measure was developed to detect whether vertical rules span the full width of pages.

5.7.10 Text Formatting Measures: Text in Clusters

As discussed above, the literature also discusses the amount of text contained in text clusters [Furnas 1997; Nielsen 2000; Schriver 1997]. Schriver [1997] suggests that Web designers minimize the amount of continuous text in clusters [Schriver 1997]. No measure was developed to assess the amount of text in clusters, since the current cluster measures are not highly accurate.

5.8 Link Formatting Measures

Table 5.10 summarizes three link formatting measures developed and discussed in this section. These measures assess the following Web interface features.

1. Are there text links that are not underlined?
2. What colors are used for links?

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
Are there text links that are not underlined?								
Non-Underlined Text Links	Whether there are text links without visible underlines	√	√	√		–	–	91.7%
What colors are used for links?								
Link Color Count	Number of colors used for text links		√	√		100.0%	99.2%	99.6%
Standard Link Color Count	Number of default browser colors used for text links		√	√		100.0%	100.0%	100.0%

Table 5.10: Summary of link formatting measures. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

5.8.1 Link Formatting Measures: Non-Underlined Links

It is possible for text links without underlines to be overlooked, since users are accustomed to the converse. Sawyer and Schroeder [2000] suggest that Web designers avoid using non-underlined text links, because they may be confused with text or considered a placeholder. A Non-Underlined Text Links measure was developed to detect the presence of text links without visible underlines.

5.8.2 Link Formatting Measures: Link Colors

The surveyed literature provides some guidance on colors used for text links [Nielsen 2000; Sawyer and Schroeder 2000; Spool *et al.* 1999], including the following.

- Use distinct link and visited link colors [Nielsen 2000; Sawyer and Schroeder 2000].
- Use link and visited link colors that are similar to default browser colors (shades of blue, red, and purple) [Nielsen 2000].
- Use default browser colors for links [Spool *et al.* 1999].

A Link Color Count was developed to assess the number of colors used for text links. No measures were developed to detect whether separate colors are used for unvisited and visited links, nor whether colors are similar to the default browser colors; such measures will be developed in future work. A Standard Link Color Count was developed to measure the number of default browser colors used (e.g., blue used for unvisited links).

One limitation of the color measures is that they may count colors that are not discriminable or perceptually close. For example, it is possible that users may consider link text colored with two different shades of blue as being the same color; color counts would be inflated in this case. Future work will consider the perceptual closeness of colors in metrics computation.

5.9 Graphic Formatting Measures

Table 5.11 summarizes the seven graphic formatting measures developed for assessing the following features of Web interfaces.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How tall are graphics?								
Minimum Graphic Height	Minimum image height			√		–	–	100.0%
Maximum Graphic Height	Maximum image height			√		–	–	100.0%
Average Graphic Height	Average image height			√		–	–	100.0%
How wide are graphics?								
Minimum Graphic Width	Minimum image width			√		–	–	100.0%
Maximum Graphic Width	Maximum image width			√		–	–	100.0%
Average Graphic Width	Average image width			√		–	–	100.0%
How much page area is covered by graphics?								
Graphic Pixels	Total page area covered by images			√		–	–	100.0%

Table 5.11: Summary of graphic formatting measures. All of the measures are expressed in pixels. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

1. How tall are graphics?
2. How wide are graphics?
3. How much page area is covered by graphics?

Several literature sources discuss the sizes of images (the number of pixels in an image) [Flanders and Willis 1998; Nielsen 2000; Schriver 1997]. Flanders and Willis [1998] suggest that Web designers avoid using large graphics, although they do not quantify what is considered a large graphic. Hence, several measures were developed to assess the sizes of images: the minimum, maximum, and average height of images; the minimum, maximum, and average width of images; and the total pixel area covered by images. The latter measure in conjunction with a measure of the pixel area required for the page (discussed in Section 5.10.8) provides insight about what portion of the page is covered by images.

5.10 Page Formatting Measures

Tables 5.12, 5.13, and 5.14 summarize the 27 page formatting measures developed and discussed in this section. These measures assess the following aspects of Web interfaces.

1. How are colors used across the page?
2. What fonts are used across the page?
3. How big is the page? Big in this context refers to the width and height of pages.
4. Are there interactive elements on the page?
5. How is the page's style controlled?
6. Information about other page characteristics (e.g., the page's functional type and self-containment). Section 5.11 discusses functional types for pages, including home, link, and form pages. Self-containment in this context refers to the degree to which the page is rendered with HTML code and images, as opposed to being rendered with stylesheets, applets, scripts, etc.

The HTML Parser and Browser Emulator (see Chapter 4) was configured to simulate the Netscape Navigator 4.75 browser with default window settings (menu bar, address bar, toolbar, and status line). Currently, most monitors are 800 x 600 pixels [DreamInk 2000]; thus, the window size was fixed at 800 x 600 pixels for computing page formatting measures.

5.10.1 Page Formatting Measures: Colors

Colors used on computer screens, in Web interfaces in particular, is discussed extensively in the literature [Ambühler and Lindenmeyer 1999; Flanders and Willis 1998; Kaiser 1998; Murch 1985; Sano 1996; Schriver 1997; Stein 1997]. Specific guidance includes the following.

- Use no more than 6 discriminable colors [Murch 1985].
- Use browser-safe colors [Kaiser 1998].
- Use 256 (i.e., 8 bit) color palettes [Sano 1996].

In addition to the color measures developed to assess text and link formatting (Sections 5.7.5 and 5.8.2), three measures were developed to assess color use at the page level. A Color Count measures the total number of unique colors used for text, links, backgrounds, table areas, etc.; this measure made significant contributions in distinguishing Web pages in both prior metric studies. For each color used, the number of times it is used on the page is also tracked. For example, for every word on the page, the corresponding color use count is updated. The Minimum Color Use measure reports the minimum number of times a color is used. The average and maximum number of times a color is used is not reported, since these measures would be proportional to the amount of text on the page. This measure detects the use of an accent or sparsely-used color.

Use of "good" colors is also assessed through several measures. The Web design literatures encourages the use of browser-safe colors or colors whose RGB values are all evenly divisible by 51 [Kaiser 1998]. The Browser-Safe Color Count reports the number of such colors. Other measures to assess the quality of color combinations were developed and are discussed below.

One limitation of the color measures is that they may count colors that are not discriminable or perceptually close. For example, it is possible that users may consider two shades of blue with distinct RGB values as being the same; color counts would be inflated in this case. Future work will consider the perceptual closeness of colors in metrics computation.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How are colors used across the page?								
Color Count	Number of colors used			√		–	–	99.5%
Minimum Color Use	Minimum number of times a color is used			√		–	–	100.0%
Browser-Safe Color Count	Number of browser-safe colors used			√		100.0%	100.0%	100.0%
Good Text Color Combination	Number of good text or thin line color combinations			√		100.0%	100.0%	100.0%
Neutral Text Color Combinations	Number of neutral text or thin line color combinations			√		100.0%	100.0%	100.0%
Bad Text Color Combinations	Number of bad text or thin line color combinations			√		100.0%	100.0%	100.0%
Good Panel Color Combinations	Number of good thick line or panel color combinations			√		100.0%	100.0%	100.0%
Neutral Panel Color Combinations	Number of neutral thick line or panel color combinations			√		100.0%	100.0%	100.0%
Bad Panel Color Combinations	Number of bad thick line or panel color combinations			√		100.0%	100.0%	100.0%

Table 5.12: Summary of page formatting measures (Table 1 of 3). The quality of text and panel color combinations is assessed based on research in [Murch 1985]. The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
What fonts are used across the page?								
Font Count	Number of fonts used			√		–	–	95.7%
Serif Font Count	Number of serif font faces used			√		–	–	100.0%
Sans Serif Font Count	Number of sans serif font faces used			√		–	–	100.0%
Undetermined Font Style Count	Number of undetermined font faces used			√		–	–	100.0%
How big is the page?								
Page Height	Height of page in pixels (600 pixel screen height)			√		–	–	88.2%
Page Width	Width of page in pixels (800 pixel screen width)			√		–	–	95.7%
Page Pixels	Total screen area required to render the page			√		–	–	84.0%
Vertical Scrolls	Number of vertical scrolls required to view the entire page				√	–	–	85.9%
Horizontal Scrolls	Number of horizontal scrolls required to view the entire page				√	–	–	100.0%
Are there interactive elements on the page?								
Interactive Element Count	Number of text fields, buttons, and other form objects				√	–	–	100.0%
Search Element Count	Number of forms for performing a search				√	–	–	91.7%

Table 5.13: Summary of page formatting measures (Table 2 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How is the page's style controlled?								
External Stylesheet Use	Whether an external stylesheet file is used to format the page				√	–	–	100.0%
Internal Stylesheet Use	Whether an internal stylesheet is used within the <head> tag to format the page				√	–	–	100.0%
Fixed Page Width Use	Whether tables are used to create a specific page width			√		–	–	100.0%
Other page characteristics (e.g., page's functional type and self-containment)								
Page Depth	Level of the page within the site (determined by (crawling order)		√			–	–	–
Page Type	The page's functional type				√	–	–	84.0%
Self Containment	The degree to which all page elements are rendered solely via the HTML and image files				√	–	–	100.0%
Spamming Use	Whether the page uses invisible text or long page titles possibly indicating spamming				√	–	–	100.0%

Table 5.14: Summary of page formatting measures (Table 3 of 3). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

5.10.2 Page Formatting Measures: Color Combinations

The quality of color combinations used on computer screens, in Web pages in particular, has also been discussed in the literature [Flanders and Willis 1998; Murch 1985; Nielsen 2000]. Specific guidance includes the following.

- Use color combinations determined to be good (i.e., high contrast) via research studies [Murch 1985].
- Avoid using black backgrounds [Flanders and Willis 1998].
- Use high contrast between background and text [Flanders and Willis 1998; Nielsen 2000].

Murch [1985] conducted a study with sixteen users wherein color combinations on computer screens were examined; combinations of white, black, red, green, blue, cyan, magenta, and yellow were used. From this study, the author developed tables summarizing foreground and background color combinations that were preferred or rejected by at least 25% of the participants. Color combinations were examined for text and thin lines (i.e., less than three pixels in width or height) as well as for thick lines and panels (i.e., large shaded areas, such as a colored navigation bar).

Based on this information, six measures were developed to report the number of good, neutral, and bad color combinations for text (text and thin lines) and panels (panels and thick lines). Color combinations are only reported as being bad or good if 40% (seven or more) of the participants rejected or preferred the color combinations in Murch's study; this is a more stringent criteria than the 25% cutoff used by Murch. The previously-discussed CIE Luv color space is used as a starting point for mapping RGB values from a Web page into one of the eight study colors. A CIE Luv color is then mapped into the closest Munsell color [Foley *et al.* 1990]; this final mapping identifies a corresponding hue (i.e., one of the eight study colors). Over 700 Munsell color swatches are used for this mapping; the mapping algorithm and Munsell color swatch information was provided courtesy of Chris Healey [Healey 1996].

The Metrics Computation Tool analyzes all foreground and background color combinations used in a Web page, provided these color combinations are embedded within the HTML code. As previously discussed, the hues identified for foreground and background colors are used to determine if the color combination is good, bad, or neutral (i.e., not rejected or preferred by at least 40% of study participants). If two colors map into the same hue, then the distance between the colors is considered to determine whether it is a bad (i.e., colors are too close) or neutral (i.e., colors are distinct enough) color combination. Experimenting with various color distances revealed that a Euclidean distance of 25 was adequate for determining if two colors are distinct enough; this distance is the square of the previously-discussed perceptual closeness distance [Jackson *et al.* 1994].

Cultural information is not taken into consideration when determining the quality of color combinations. Whether the page background is black is also not reported. Color combinations used in images, applets, etc. are not assessed, since this assessment requires image processing techniques.

One limitation of the color combination measures is that they may count color combinations that are not discriminable or perceptually close. For example, it is possible that users may consider two areas with white foreground text and backgrounds that are shades of blue with distinct RGB values as being the same; color combination counts would be inflated in this case. Future work will consider the perceptual closeness of colors in metrics computation. Another limitation is that the color palette is not evaluated to determine if a good selection of colors is used; this will be explored in future work.

5.10.3 Page Formatting Measures: Fonts

A font is a combination of four features: a font face, a font size, whether text is bolded, and whether text is italicized [Schriver 1997]. Several sources discuss fonts [Nielsen 2000; Schriver 1997; Stein 1997], and Nielsen [2000] suggests that Web designers use no more than two fonts and possibly one for special text. A Font Count was developed to report the total number of unique fonts used on a page; this measure played a significant role in distinguishing highly-rated pages in the second metrics study. The number of each type of font face – serif, sans serif, and undetermined font styles – is also reported. The measures do not report whether a font or font face is used for special text nor do they distinguish fonts used for body and display text.

The font measures in this section are distinct from the ones in Section 5.7.3. The latter ones assess the number of words formatted with serif and sans serif font styles; thus, proving some insight about how fonts are used.

5.10.4 Page Formatting Measures: Line Length

The width of text lines on the page is discussed in the literature [Flanders and Willis 1998; Schriver 1997], and specific guidance includes the following.

- Keep line lengths to 40–60 characters [Schriver 1997].
- Keep text between 9 to 15 words per line [Flanders and Willis 1998].

No measure was developed to assess text line lengths, since lengths vary considerably when multiple columns are used. Image processing techniques are required to accurately determine line lengths.

5.10.5 Page Formatting Measures: Leading

Spacing between consecutive text lines on a page is more of a concern for print documents than Web documents, since Web pages typically use consistent spacing as dictated by browsers. Stylesheet parameters can also control leading. Schriver [1997] suggests that leading be 120% of the font face's point size and even larger between paragraphs. Given that leading is mainly controlled by the browser, no measure was developed to assess this feature.

5.10.6 Page Formatting Measures: Framesets

Use of framesets is an often debated topic in Web design literature, since they typically confuse users [Flanders and Willis 1998; Nielsen 2000; Stein 1997]. Specific guidance includes the following.

- Avoid using framesets [Nielsen 2000].
- Use tables instead of framesets [Flanders and Willis 1998].

Although simple to measure, no measure was developed to report the use of framesets, since the tool was not designed to support them. However, the tool does support inline frames, since text equivalents are typically provided by designers; inline frames are only supported by the Internet Explorer browser. All metric studies, including the study discussed in the next chapter, excluded sites that used framesets.

5.10.7 Page Formatting Measures: Interactive Elements

Use of buttons, text boxes, pull-down menus, and other interactive elements has been discussed extensively in the literature [Flanders and Willis 1998; Rosenfeld and Morville 1998; Sawyer and Schroeder 2000; Scanlon and Schroeder 2000b; Spool *et al.* 1999]. Specific guidance on interactive elements includes the following.

- Avoid using mouseovers and pull-down menus for navigation [Rosenfeld and Morville 1998; Sawyer and Schroeder 2000].
- Support search; users use search half of the time [Scanlon and Schroeder 2000b].
- Make the scope and results of searching clear [Spool *et al.* 1999].
- Do not use form buttons as links (i.e., overuse buttons) [Flanders and Willis 1998].

An Interactive Element Count was developed to report the total number of form elements used (all buttons, select menus, checkboxes, etc.). A Search Element Count was also developed to determine whether searching is supported on the page. Several heuristics were developed to detect search forms with 92% overall accuracy. For example, if the form or some element in the form contains the term "search," then it is considered a search form. One limitation of the measure is that it does not detect whether the search is for the site itself or for the Web at large.

No measures were developed to detect the use of form elements for navigation or whether the scope and results of searching are clear.

5.10.8 Page Formatting Measures: Screen Size

Much guidance is provided in the literature on the width and height of Web interfaces [Flanders and Willis 1998; Nielsen 2000; Sano 1996; Sawyer *et al.* 2000], including the following.

- Limit horizontal width to 572 pixels or less [Sano 1996].
- Restrict page width and height to 595 x 295 pixels [Flanders and Willis 1998].
- Restrict page width to less than 600 pixels [Nielsen 2000].
- Longer pages are better; use 800 x 600 pixels; and avoid horizontal scrolling [Sawyer *et al.* 2000].

Differences in page width and height recommendations reflect changes in the dominant screen sizes over the years. Currently, most monitors are 800 x 600 pixels [DreamInk 2000]. Thus, measures of the width and height of a page were developed using an 800 x 600 screen size. The available width for pages is determined based on the Netscape Navigator 4.75 browser; small areas are occupied on the left and right of the browser window for a scroll bar and border. A Page Pixels measure was developed to report the total number of pixels covered by the page; this measure is the page width multiplied by the page height.

5.10.9 Page Formatting Measures: Screen Coverage

The total screen area covered (i.e., non whitespace) is also discussed in the literature [Sawyer *et al.* 2000; Schriver 1997; Spool *et al.* 1999]. The consensus is that Web designers minimize whitespace on the page [Sawyer *et al.* 2000; Spool *et al.* 1999]. No measure was developed to determine the total screen area covered, since this computation may require image processing techniques. For example, blank images are often used extensively as spacers; counting areas covered by spacers could inflate screen coverage.

5.10.10 Page Formatting Measures: Text Density

The screen area covered by text is discussed in the literature [Sawyer *et al.* 2000; Schriver 1997; Spool *et al.* 1999]; specific guidance includes the following.

- Text should cover no more than 25–30% of the screen area [Schriver 1997].
- Greater text density facilitates page scanning [Sawyer *et al.* 2000; Spool *et al.* 1999].

These guidelines obviously contradict each other. However, no measure was developed to determine the text density, since this computation may require image processing techniques to compute accurately.

5.10.11 Page Formatting Measures: Scrolling

Vertical and horizontal scrolling is discussed extensively in the literature [Flanders and Willis 1998; Nielsen 2000; Spool *et al.* 1999]; specific guidance includes the following.

- Minimize scrolling [Spool *et al.* 1999].
- Minimize vertical scrolling to 2 screens [Flanders and Willis 1998].
- Users should not be required to scroll [Nielsen 2000].

The number of vertical and horizontal scrolls required to view the entire page is measured. An 800 x 600 screen size is used for these computations, since most monitors are at least 800 x 600 pixels [DreamInk 2000]. The available width for pages is determined based on the Netscape Navigator 4.75 browser; small areas are occupied on the left and right of the browser window for a scroll bar and border. Similarly, the available height is adjusted to account for the Netscape default menu bar, toolbar, address bar, and status bar.

5.10.12 Page Formatting Measures: Stylesheets

Use of stylesheets to control page layout has been discussed in the literature [Flanders and Willis 1998; Nielsen 2000], and specific recommendations include the following.

- Use stylesheets to enforce consistency [Flanders and Willis 1998; Nielsen 2000].
- Use external rather than embedded stylesheets [Nielsen 2000].

Two measures – External and Internal Stylesheet Use – were developed to detect the use of external stylesheet files and internal stylesheets within the <head> tag of Web pages, respectively. Both of these measures are associated with experience design in Table 5.14, since the affects of stylesheets are more overarching than just information, navigation, and graphic design. For instance, use of external and internal stylesheets can potentially improve consistency across pages and reduce download time. Use of embedded style tags (i.e., style attributes within HTML tags or within the body of Web pages) is not detected, since this is not considered as a consistent use of stylesheets to control page layout. A related measure – Fixed Page Width – detects whether tables are used to force a fixed page width or not.

5.10.13 Page Formatting Measures: Layout Quality

All of the measures discussed in this section provide some insight about some aspects of layout quality. Other high-level aspects discussed in the literature include: aesthetics, alignment, balance, and the presence of distractions (e.g., popup windows or spawning browser windows) [Flanders and Willis 1998; Nielsen 2000; Sano 1996; Scanlon and Schroeder 2000a; Schriver 1997]. High-level features such as aesthetics are difficult to measure in an automated manner, since this is largely subjective and requires input from users. Other aspects, such as alignment and balance, require image processing techniques to assess; hence, no measures were implemented to assess them. Although it is suggested that Web designers minimize distractions (e.g., spawning browser windows) [Scanlon and Schroeder 2000a], the presence of distractions is not detected, since these elements are typically implemented with scripting.

Other features associated with layout quality, such as page depth, self-containment, and spamming use are measured. The page depth is determined by the crawling order within a site, and does not necessarily reflect the actual depth of the page in the site. Self-containment reflects the degree to which all elements required to render the page are contained in the HTML page itself and image files, as opposed to being rendered by external stylesheet files, scripts, applets, and other objects. For example, if a page does not use external stylesheets, scripts, applets, etc., then it is considered to have high self-containment. If long page titles (more than 64 characters) or a series of invisible words are in the page, then spamming use is reported for the page.

5.11 Page Function

Another aspect of page formatting is the primary function of Web pages (i.e., whether pages are primarily for content or links), which needs to be considered during design [Flanders and Willis 1998; Sano 1996; Stein 1997]. The first metric study showed that considering page function (home vs. other pages) leads to more accurate predictions. Although several studies have focused on automatically predicting the genre of a site (e.g., commercial or academic) [Bauer and Scharl 2000; Hoffman *et al.* 1995], few studies have been conducted to determine ways for automatically predicting the function of a page [Karlgren 2000; Pirolli *et al.* 1996; Stein 1997]. Stein [1997] considers the ratio of text link words to all of the words on a page. If this ratio is high, then the page is considered to be primarily for links; no guidance is given for what constitutes a high ratio.

Karlgren [2000] surveyed over 600 Web users to determine a set of eleven genres (e.g., home pages, searchable indices, journalistic, reports, FAQs, and others) and used 40 linguistic (e.g., number of adverbs, characters, long words, present participles, etc.) and Web (e.g., number of images and links) measures for predicting page type. Karlgren developed a decision tree to generate predictions; however, the author did not report the accuracy of the tree nor provide model details. This work was incorporated into a search interface that clusters search results by genre.

Pirolli *et al.* [1996] present a set of functional categories for Web pages, including home (organizational and personal), index, source index (sub-site home pages), reference, destination (sink pages such as acronym, copyright, and bibliographic references), and content. Pages can belong to multiple categories in their classification scheme. The authors developed a classification algorithm based on features that a Web page exhibits (e.g., page size, number of inbound and outbound links, depth of children, and similarity to children). For example, reference pages had a relatively larger page size and fewer inbound and outbound links. Page classification was used in conjunction with text similarity, hyperlink structure, and usage data to predict the interests (i.e., plausible information-seeking goals) of Web site users.

The work in [Karlgren 2000] and [Pirolli *et al.* 1996] was used as a starting point for developing an approach for predicting page types. Examination of the eleven genres in [Karlgren 2000] revealed that most qualified the type of text on the page (e.g., journalistic, discussions, lists, and FAQs), while the remaining genres were for pages with different functionality (e.g., links, forms, and home pages). Similarly, the categories in [Pirolli *et al.* 1996] contain multiple text, link, and home page types. Hence, these categories were collapsed into the following five page types.

Home: main entry pages to a site that typically provide a broad overview of site contents.

Link: pages that mainly provide one or more lists of links. Links may be annotated with text or grouped with headings (e.g., yahoo directory, redirect page, or sitemap). This functional type includes category pages (entry pages to sub-sites or major content areas).

Content: pages that mainly provide text. This functional type includes reference (e.g., a glossary, FAQ, search and site tips, and acronyms) and legal (e.g., disclaimers, privacy statements, terms, policies, and copyright notices) pages.

Form: pages that are primarily HTML forms.

Other: all remaining graphical (e.g., splash pages, image maps, and Flash) and non-graphical (e.g., blank, under construction, error, applets, text-based forms, and redirect) pages.

With the assistance of an undergraduate student, Deep Debroy, a sample of 1,770 Web pages were classified into the five categories above; each page was assigned to only one category. The pages came from 6 types of sites (Community, Education, Finance, etc.) with low to high expert ratings (see Chapter 6); there were at least 223 pages for each functional type. Multiple Linear Regression analysis [Keppel and Zedeck 1989] was used to identify a subset of the measures for model building. The Classification and Regression Tree (C&RT) [Breiman *et al.* 1984] method with 70% training and 30% test samples was used on this subset of measures. The resulting tree contains 70 rules and has an accuracy of 87% and 75% for the training and test samples, respectively. Cross validation [Schaffer 1993] was also used to further assess whether the model is generalizable to new data; the tree has a 5-fold cross validation accuracy of 75%. Figures 5.3 and 5.4 present example rules for predicting each page type.

5.12 Page Performance Measures

Tables 5.15–5.19 summarize 37 page performance measures developed to answer the following questions.

1. How fast is the page rendered?

if ((All Page Text Score is not missing AND (All Page Text Score $\leq$ 10.5)) AND (HTML Bytes is missing OR (HTML Bytes $>$ 2735)) AND (Link Count is missing OR (Link Count $>$ 8.5)) AND (Meta Tag Word Count is missing OR (Meta Tag Word Count $>$ 18.5)))

PageType = Home

This rule classifies pages as home pages if they have: minimal content similarity between source and destination page text (good visible and invisible words; this is typically because home pages are the first pages crawled and are therefore not compared to a source page); an HTML page size of more than 2.7K; 8.5 or more links; and 18.5 or more meta tag words.

---

if ((All Page Text Score is missing OR (All Page Text Score $>$ 10.5)) AND (Good Body Word Count is not missing AND (Good Body Word Count $\leq$ 86.5)) AND (Link Word Count is missing OR (Link Word Count $\leq$ 73)) AND (Good Body Word Count is not missing AND (Good Body Word Count $\leq$ 22.5)) AND (Link Count is not missing AND (Link Count $>$ 21.5)) AND (Script Bytes is missing OR (Script Bytes $\leq$ 2678.5)) AND (Interactive Object Count is missing OR (Interactive Object Count $\leq$ 1.5)))

PageType = Link

This rule classifies pages as link pages if they have: some content similarity between source and destination page text (good visible and invisible words); between 22.5 and 86.5 good body words; 73 or fewer link words; 1.5 or fewer interactive objects; more than 21.5 links; and less than 2.7K of script bytes (possibly for form validation).

---

if ((All Page Text Score is missing OR (All Page Text Score $>$ 10.5)) AND (Link Word Count is not missing AND (Link Word Count $>$ 77.5)) AND (Good Body Word Count is not missing AND (Good Body Word Count $>$ 278)) AND (Exclaimed Body Word Count is not missing AND (Exclaimed Body Word Count $\leq$ 0.5)))

PageType = Content

This rule classifies pages as content pages if they have: some content similarity between source and destination page text (good visible and invisible words); more than 77.5 link words; more than 278 good body words; and virtually no body words that are near exclamation points.

Figure 5.3: Example decision tree rules for predicting page types (Home, Link, and Content pages).

if ((All Page Text Score is not missing AND (All Page Text Score $\leq$ 10.5)) AND (HTML Bytes is not missing AND (HTML Bytes $\leq$ 2735)) AND (Search Object Count is not missing AND (Search Object Count $>$ 0.5)))

PageType = Form

This rule classifies pages as form pages if they have: minimal content similarity between source and destination page text; 2.7K or fewer HTML bytes; and search support.

---

if ((All Page Text Score is not missing AND (All Page Text Score $\leq$ 10.5)) AND (HTML Bytes is not missing AND (HTML Bytes $\leq$ 2735)) AND (Search Object Count is missing OR (Search Object Count $\leq$ 0.5)))

PageType = Other

This rule classifies pages as other pages if they have: minimal content similarity between source and destination page text; 2.7K or fewer HTML bytes; and virtually no search support.

Figure 5.4: Example decision tree rules for predicting page types (Form and Other pages).

2. Is the page accessible to people with disabilities?
3. Are there HTML errors on the page?
4. Is there strong "scent" to the page? Scent in this context refers to whether the page text provides hints about the contents on a linked page without having to navigate to the page.

5.12.1 Page Performance Measures: Page Bytes

The total bytes for the Web page (HTML code, stylesheets, objects, and images) plays an important role in how fast the page loads in the browser [Flanders and Willis 1998; Nielsen 2000]. Nielsen [2000] recommends that Web designers keep page bytes below 34K for fast loading. Most HTML authoring tools, such as Microsoft FrontPage and Macromedia Dreamweaver, provide download speed estimates; these estimates typically only consider the total bytes for pages. Experiments using only the total page bytes to estimate download speed revealed that such estimates only account for roughly 50% of download speed. Hence, a more accurate model of download speed was developed; this model is discussed in Section 5.12.4.

A Page Bytes measure was developed and shown to make significant contributions in distinguishing Web pages in both prior metric studies. The percentage of page bytes attributable to graphics was also shown to play a significant role in distinguishing highly-rated pages in the second study. These measures are not used in the download speed computation, since different types of bytes (e.g., graphic bytes or script bytes) influence download speed differently. The page bytes measure does not capture bytes that may be embedded in objects, stylesheets, scripts, etc. For example, if a stylesheet imports a secondary stylesheet, page bytes for the first stylesheet are counted but not for the second one.

The page bytes and graphic percentage measures were replaced with an HTML bytes measure – total bytes for HTML tags, excluding tags for scripts, layers, and other objects – and the number of HTML files. Similar measures were developed for graphics, scripts, and objects as discussed in the sections below.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How fast is the page rendered?								
Table Count	Number of HTML tables used to render the page				√	–	–	100.0%
HTML File Count	Number of HTML files, including stylesheet files				√	100.0%	100.0%	100.0%
HTML Bytes	Total bytes for HTML tags, text, and stylesheet tags				√	100.0%	100.0%	100.0%
Graphic File Count	Number of image files				√	100.0%	100.0%	100.0%
Graphic Bytes	Total bytes for image files				√	100.0%	100.0%	100.0%
Script File Count	Number of script files				√	100.0%	100.0%	100.0%
Script Bytes	Total bytes for scripts (embedded in script tags and in script files)				√	100.0%	100.0%	100.0%
Object File Count	Number of object files (e.g., for applets, layers, sound, etc.)				√	100.0%	100.0%	100.0%
Object Bytes	Total bytes for object tags and object files				√	100.0%	100.0%	100.0%
Object Count	Number of scripts, applets, objects, etc.				√	–	–	100.0%
Download Time	Time for a page to fully load over a 56.6K modem (41.2K connection speed)				√	–	–	86.0%

Table 5.15: Summary of page performance measures (Table 1 of 5). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
Is the page accessible to people with disabilities?								
Bobby Approved	Whether the page was approved by Bobby as being accessible to people with disabilities				√	–	–	100.0%
Bobby Priority 1 Errors	Number of Bobby priority 1 errors reported				√	–	–	100.0%
Bobby Priority 2 Errors	Number of Bobby priority 2 errors reported				√	–	–	100.0%
Bobby Priority 3 Errors	Number of Bobby priority 3 errors reported				√	–	–	100.0%
Bobby Browser Errors	Number of Bobby browser compatibility errors reported				√	–	–	100.0%
Are there HTML errors in the page?								
Weblint Errors	Number of HTML syntax errors reported by Weblint				√	–	–	100.0%
Is there strong "scent" to the page?								
Visible Page Text Terms	Maximum good visible words on source & destination pages	√	√			–	–	100.0%
Visible Unique Page Text Terms	Maximum good visible, unique words on source & destination pages	√	√			100.0%	100.0%	100.0%
Visible Page Text Hits	Common visible words on source & destination pages	√	√			100.0%	100.0%	100.0%

Table 5.16: Summary of page performance measures (Table 2 of 5). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
Is there strong "scent" to the page?								
Visible Page Text Score	Score for common visible words on source & destination pages	√	√			100.0%	100.0%	100.0%
All Page Text Terms	Maximum good visible and invisible words on source & destination pages	√	√			–	–	100.0%
All Unique Page Text Terms	Maximum good visible and invisible, unique words on source & destination pages	√	√			100.0%	100.0%	100.0%
All Page Text Hits	Common visible and invisible words on source & destination pages	√	√			100.0%	100.0%	100.0%
All Page Text Score	Score for common visible and invisible words on source & destination pages	√	√			100.0%	100.0%	100.0%
Visible Link Text Terms	Maximum good visible words in the link text & destination page	√	√			–	–	100.0%
Visible Unique Link Text Terms	Maximum good visible, unique words in the link text & destination page	√	√			100.0%	100.0%	100.0%
Visible Link Text Hits	Common visible words in the link text & destination page	√	√			100.0%	100.0%	100.0%

Table 5.17: Summary of page performance measures (Table 3 of 5). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
Is there strong "scent" to the page?								
Visible Link Text Score	Score for common visible words in the link text & destination page	√	√			100.0%	100.0%	100.0%
All Link Text Terms	Maximum good visible and invisible words in the link text & destination page	√	√			–	–	100.0%
All Unique Link Text Terms	Maximum good visible and invisible, unique words in the link text & destination page	√	√			100.0%	100.0%	100.0%
All Link Text Hits	Common visible and invisible words in the link text & destination page	√	√			100.0%	100.0%	100.0%
All Link Text Score	Score for common visible and invisible words in the link text & destination page	√	√			100.0%	100.0%	100.0%
Page Title Terms	Maximum good page title words on source & destination pages	√	√			–	–	100.0%
Unique Page Title Terms	Maximum good, unique page title words on source & destination pages	√	√			100.0%	100.0%	100.0%

Table 5.18: Summary of page performance measures (Table 4 of 5). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
Is there strong "scent" to the page?								
Page Title Hits	Common page title words on source & destination pages	√	√			100.0%	100.0%	100.0%
Page Title Score	Score for common page title words on source & destination pages	√	√			100.0%	100.0%	100.0%

Table 5.19: Summary of page performance measures (Table 5 of 5). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

5.12.2 Page Performance Measures: Graphic Bytes

The literature discusses the total bytes for images on the page [Flanders and Willis 1998; Nielsen 2000]. Specifically, Flanders and Willis [1998] suggest that Web designers keep graphic bytes to less than 35K and to optimize images. This recommendation obviously contradicts the recommendation from Nielsen to keep the Web page and all elements under 34K [Nielsen 2000]. A graphic bytes measure was developed to report this information. The number of graphic files is also reported.

5.12.3 Page Performance Measures: Objects

Use of applets, controls, scripts, marquees, plug-ins, etc. is discussed extensively in the literature [Ambühler and Lindenmeyer 1999; Flanders and Willis 1998; Nielsen 2000; Rosenfeld and Morville 1998; Spool *et al.* 1999; Stein 1997]; specific guidance includes the following.

- Avoid gratuitous use of technology [Nielsen 2000; Rosenfeld and Morville 1998].
- Minimize the use of video [Nielsen 2000].
- Avoid using sound files [Flanders and Willis 1998].

Separate measures were developed to track the number of script and layer files as well as the total bytes for these elements, including bytes for the HTML tags within the Web page. Similarly, the number of object files and the total bytes for objects is measured. These measures are used in conjunction with previously-discussed measures to estimate the download speed (see discussion below). The number of bytes and files do not capture bytes or files that may be embedded in objects and scripts. For example, if an object loads images, then the bytes for the loaded images are not counted although bytes for the object itself are.

The total number of objects – scripts (both scripts within the HTML page and in an external script file), layers, applets, etc. – is also reported.

5.12.4 Page Performance Measures: Download Speed

The time for a page to fully load is considered a critical issue for Web interfaces [Flanders and Willis 1998; Nielsen 2000; Scanlon and Schroeder 2000a; Spool *et al.* 1999; Zona Research 1999]. Specific guidance includes the following.

- Download speed should be no more than 10 seconds [Nielsen 2000].
- Home pages greater than 40K result in significant bailouts [Zona Research 1999].
- Perceived download speed matters more than actual download speed [Scanlon and Schroeder 2000a].

Many HTML authoring tools provide download speed estimates to assist Web designers with optimizing pages. Experience with these estimates on the sample revealed that such estimates only account for about 50% of the actual download speed; there is also a major discrepancy for download speed estimates provided by the Bobby tool [Clark and Dardailler 1999; Cooper 1999]. These discrepancies are mostly due to the fact that these tools use a simplistic model, typically based on the total bytes for pages and in some cases, such as Bobby, incorporate a latency measure proportional to the total number of files.

A model of download speed was developed based on the actual download speeds for the sample. A 56.6K modem was used to measure actual download speeds, since this is currently the most prevalent modem [DreamInk 2000]. The measurement activity revealed that it is rarely possible to achieve a 56.6K connection speed with a 56.6K modem; this is due to various technological limitations of the analog modem [Bash 1997] and possibly poor telephone connections. For 50 connection sessions to three different Internet service providers at various times of the day, the average and median connection speed were 41.2K with 42.6K as a close second. Hence, a connection speed of 41.2K was used to measure download speed for the sample; this value is also used for estimating download speed. All fourteen of the Web pages were downloaded twelve times each. Several warm-up downloads were performed and browser caching was disabled during this measurement activity.

Several measures believed to impact download speed were developed: Graphic Bytes and Graphic File Count (Section 5.12.2), HTML Bytes and HTML File Count (Section 5.12.1), Script Bytes and Script File Count (Section 5.12.3), Object Bytes and Object File Count (Section 5.12.3), Object Count (Section 5.12.3), and Table Count (the number of <table> tags in the Web page). These measures and the actual download speeds were used for Multiple Linear Regression analysis [Keppel and Zedeck 1989] to determine an equation for predicting download speed. A backward elimination method was used wherein all of the measures were entered into the equation initially, and then one by one, the least predictive measure was eliminated. This process was repeated until the Adjusted R Square (predictive accuracy) showed a significant increase with the elimination of a predictor. All of the measures were retained by this method, except for the Object Count.

Equation 5.2 depicts the model developed to predict the download speed for a Web page. The Adjusted R Square for Equation 5.2 is .86 indicating that the measures explained about 86% of the variance in predictions. The F value is 123.06 and significant at the $p < .01$ level; this indicates that the linear combination of the measures significantly predicts the download speed. Table 5.20 shows the relative contribution of each measure to download speed predictions; all measure contributions are significant at the $p < .01$ level.

$$DownloadSpeed = -0.181 + 0.0003085 * GraphicBytes + \quad (5.2)$$

$$
\begin{aligned}
&0.0002492 * HTML\,Bytes + \\
&0.0005748 * Script\,Bytes + \\
&0.003211 * Object\,Bytes + \\
&10.140 * HTML\,File\,Count + \\
&-0.385 * Graphic\,File\,Count + \\
&-1.378 * Script\,File\,Count + \\
&-6.243 * Object\,File\,Count + \\
&-0.463 * Table\,Count
\end{aligned}
$$

5.12.5 Page Performance Measures: Accessibility

Whether a Web interface is accessible to people with disabilities is discussed in the literature [Clark and Dardailler 1999; Cooper 1999; Nielsen 2000; Web Accessibility Initiative 1999]. The consensus is that Web designers adhere to accessibility principles to create sites that serve a broad user community. In other words, accessible interfaces are of higher quality than non-accessible ones, although this claim has not been empirically examined.

Results of running the Bobby tool (version 3.2) [Clark and Dardailler 1999; Cooper 1999] is reported for each Web page; default options are used. Specifically, whether the page is Bobby approved, the number of priority 1, 2, and 3 errors, as well as the number of browser compatibility errors is reported.

5.12.6 Page Performance Measures: HTML Errors

Whether Web interfaces contain HTML or browser-incompatibility errors is also discussed in the literature [Bowers 1996; Kim and Fogg 1999; Fogg *et al.* 2000]. One survey showed that HTML errors decrease credibility [Kim and Fogg 1999; Fogg *et al.* 2000]. Hence, the results of running the Weblint tool (version 1.02) [Bowers 1996], specifically the total number of Weblint errors is reported. Weblint detects numerous HTML errors, such as missing closing tags, broken links, and invalid tag attributes. For measurement purposes, Weblint is configured to support Netscape extensions and to disable checks for minor HTML errors, such as the use of quotations around attribute values and specifying alternative text for images. The specific command line used is below.

Measure	Standardized Coefficient	Significance
Graphic Bytes	1.011	0.000
HTML Bytes	0.646	0.000
Script Bytes	0.164	0.005
Object Bytes	0.396	0.000
Graphic File Count	-0.559	0.000
HTML File Count	0.380	0.000
Script File Count	-0.184	0.001
Object File Count	-0.236	0.004
Table Count	-0.555	0.000

Table 5.20: Standardized coefficients indicating the contribution of each measure to download speed predictions along with t-test results (2-tailed significance).

weblint -x Netscape -i -d quote-attribute-value,extension-attribute,extension-markup,img-alt,attribute-delimiter filename

5.12.7 Page Performance Measures: Scent Quality

Much has been said about the use of "scent" – hints to help the user decide where to go next – as a way to improve navigation [Chi *et al.* 2000; Furnas 1997; Larson and Czerwinski 1998; Miller and Remington 2000; Nielsen 2000; Rosenfeld and Morville 1998; Sawyer *et al.* 2000; Spool *et al.* 1999; Spool *et al.* 2000]. Specific guidance includes the following.

- Use clear headings with related links (i.e., link clustering) to enhance scent [Spool *et al.* 2000].
- Effective navigation requires small pages (views), few clicks between pages, and strong scent [Furnas 1997].
- Weak scent (i.e., ambiguous link text) impedes navigation [Chi *et al.* 2000; Miller and Remington 2000; Spool *et al.* 1999; Spool *et al.* 2000].
- Avoid using 'Click Here' for link text [Nielsen 2000].
- Use breadcrumbs (i.e., displaying a navigation trail) rather than long navigation bars [Nielsen 2000].
- Use link titles to help users predict what will happen if they follow a link [Nielsen 1998b].

Many of these suggestions are difficult to measure in an automated manner. Furthermore, it is difficult to gauge users' understanding of link text. Nonetheless, several research efforts have yielded computational models of scent [Chi *et al.* 2000; Chi *et al.* 2001; Miller and Remington 2000; Pirolli and Card 1995; Pirolli *et al.* 1996; Pirolli 1997]. For Web navigation, the most promising approach compares proximal cues on the source page (link text, text surrounding the link, graphics related to a link, and the position of the link on the page) to text on the destination page [Chi *et al.* 2000; Chi *et al.* 2001; Pirolli *et al.* 1996]; this approach also considers the site's linkage topology and usage patterns from server log data. The authors have used this approach for predicting how users will navigate a site given some information need as well as for inferring users' information needs from navigation patterns. Several information retrieval approaches, such as TF.IDF (Term Frequency by Inverse Document Frequency) and simple word overlap [Baeza-Yates and Ribeiro-Neto 1999], have been used for determining similarity between the link text and destination page text.

The approach discussed in [Chi *et al.* 2000; Chi *et al.* 2001; Pirolli *et al.* 1996] was used as a starting point for developing eighteen scent quality measures. Given the nature of the Metrics Computation Tool, it was not possible to incorporate site topology and usage data into these measures; this would require exhaustive site crawling and access to server logs. Furthermore, simple word overlap is used as opposed to TF.IDF or other information retrieval algorithms, since it does not require a large collection of pages for each site. The measures focus on assessing content similarity as follows.

Source Page Text vs. Destination Page Text: This comparison establishes whether or not the content is similar between the two pages. If there is a high similarity, one would expect the link text similarity to reflect this.

Link Text vs. Destination Page Text: Link text includes the actual words in the link (including image alt attributes for graphic links) as well as twelve good words before and after the link text; if an image precedes or follows a link, text is extracted from the image's alt attribute if specified. If there is a high similarity between text on the source and destination pages, then this comparison should reflect this similarity. If not, this may indicate poor scent quality.

Source Page Title vs. Destination Page Title: This comparison is mainly to assess whether page titles vary between pages in the site. As discussed in Section 5.4.2, Nielsen [2000] suggests that designers use different page titles for each page. A similar page title consistency measure was developed to assess variation in page titles throughout the site; this measure will be discussed in Section 5.13.1.

For the first two comparisons, two forms of page text are considered: visible words (page title, headings, and body text) and all words (page title, headings, body text, meta tags, invisible text, and image alt text). The first type reflects content that can be seen by users during browsing, while the latter type reflects text used for searching, specifically indexing documents for searching.

A Java program adapted from one developed by Marti Hearst is used for the comparisons. The program employs simple word overlap and reports several measures for each comparison: 1. the maximum number of terms (good words; Section 5.4.12) considered between the source and destination text (link, title, or page contents); 2. the number of unique terms in the source text (link, title, or page contents); 3. the number of terms that appear in both the source and destination text (hits); and 4. the weighted score for the common terms. For the weighted score, each term in the source text is assigned a weight equal to the number of times it appeared in the source text; frequently-used terms will have a higher weight. When the term appears in the destination text, this weighted score is used for each hit. The actual word overlap is a ratio of these measures ($\frac{hits}{unique\ terms}$ and $\frac{score}{terms}$) and as such is not considered. The terms and unique terms are reported simply for reference. The eighteen scent quality measures are summarized below; these measures are associated with the destination page as determined by site crawling order.

Source Page Text vs. Destination Page Text Measures: visible text (browsing): terms, unique terms, hits, and score; and all text (searching): terms, unique terms, hits, and score.

Link Text vs. Destination Page Text Measures: visible text (browsing): terms, unique terms, hits, and score; and all text (searching): terms, unique terms, hits, and score.

Source Page Title vs. Destination Page Title Measures: all text (browsing and searching): terms, unique terms, hits, and score.

There are many ways to express similar concepts; however, term expansion is not used. Another major limitation of these measures is that they do not reflect subjective preferences or whether users understand the terms. Although the measures are computed with high accuracy, this is not a reflection of how accurately they align with users' perceptions. Ideally, a controlled study focusing on content similarity would be conducted. The study would capture user ratings of the similarity between link text and text on destination pages. Study results could then be used to develop better measures of scent quality. This will be addressed with future work.

One major limitation of the link text measures is that the proximity of the associated link text (i.e., twelve good words before and after a link) is not considered. For example, it is possible for associated link text to be taken from a different paragraph or even a navigation bar, since the order that text appears in the HTML is used. Image processing is needed to ensure that the associated link text is actually part of the content surrounding a link.

5.13 Site Architecture Measures

Tables 5.21 and 5.22 summarize sixteen site architecture measures for assessing the following aspects of Web interfaces.

1. How consistent are page elements?
2. How consistent is the formatting of page elements?
3. How consistent is page formatting?
4. How consistent is page performance?
5. What is the overall consistency?
6. How big is the site? Big in this context refers to the breadth and depth of pages as well as the total number of pages based on crawling with the Site Crawler Tool.

5.13.1 Site Architecture Measures: Consistency

The consistency of page layout across the site has been discussed extensively in the literature [Flanders and Willis 1998; Fleming 1998; Nielsen 2000; Mahajan and Shneiderman 1997; Sano 1996; Sawyer *et al.* 2000]. Specific guidance includes the following.

- Consistent layout of graphical interfaces result in a 10–25% speedup in performance [Mahajan and Shneiderman 1997].
- Use consistent navigational elements [Flanders and Willis 1998; Fleming 1998].
- Use several layouts (e.g., one for each page type) for variation within the site [Sano 1996].
- Consistent elements become invisible [Sawyer *et al.* 2000].

These guidelines are obviously contradictory; hence, several measures for assessing site consistency were developed. Specifically, the variation in text elements and formatting, page formatting, page performance, and other aspects are computed. Variation is an inverse measure of consistency; larger variation indicates less consistency and vice versa. The variation measures are based on the Web site aspects presented in Figure 5.5. The average variation for measures within each block of Figure 5.5 is computed, as well as overall element variation (across the bottom row of measures), overall formatting variation (2nd and 3rd row), and overall variation (all rows except the top one). Recall that the scent quality measures include the page title hits and score to assess the similarity of page titles between pairs of pages (see Section 5.12.7); variation in these measures is also computed. The following process was followed in developing these site consistency measures.

1. Compute the Coefficient of Variation (CoV, $100 * \frac{\sigma}{\bar{x}}$, where σ is the standard deviation, and $\bar{x}$ is the mean) [Easton and McColl 1997] for each measure across all of the pages in a site. The CoV is a standard, unitless measure of the amount of variance in a data set.
2. Compute the median CoV for relevant measures within a category (text element, graphic formatting, and so on). The median does not require data that is normally distributed with equal variances; thus, it is more appropriate in this case than the mean. The median CoV is reported as the variation measure (i.e., text element variation, graphic formatting variation, etc.).

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
How consistent are page elements?								
Text Element Variation	Variation in text elements across pages				√	–	–	100.0%
Page Title Variation	Variation in page titles across pages		√		√	–	–	100.0%
Link Element Variation	Variation in link elements across pages				√	–	–	100.0%
Graphic Element Variation	Variation in graphic elements across pages				√	–	–	100.0%
How consistent is formatting of page elements?								
Text Formatting Variation	Variation in text formatting across pages				√	–	–	100.0%
Link Formatting Variation	Variation in link formatting across pages				√	–	–	100.0%
Graphic Formatting Variation	Variation in graphic formatting across pages				√	–	–	100.0%
How consistent is page formatting?								
Page Formatting Variation	Variation in page formatting across pages				√	–	–	100.0%
How consistent is page performance?								
Page Performance Variation	Variation in page performance across pages				√	–	–	100.0%
What is the overall consistency?								
Overall Element Variation	Variation in all elements across pages				√	–	–	100.0%
Overall Formatting Variation	Variation in all formatting across pages				√	–	–	100.0%

Table 5.21: Summary of site architecture measures (Table 1 of 2). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

Measure	Description	Aspects Assessed				Accuracy		
		ID	ND	GD	ED	Hit	Miss	Avg.
What is the overall consistency?								
Overall Variation	Variation in elements, formatting, and performance across pages				√	–	–	100.0%
How big is the site?								
Page Count	Number of crawled pages				√	–	–	100.0%
Maximum Page Depth	Maximum crawl depth				√	–	–	100.0%
Maximum Page Breadth	Maximum pages crawled at a level				√	–	–	100.0%
Median Page Breadth	Median pages crawled across levels				√	–	–	100.0%

Table 5.22: Summary of site architecture measures (Table 2 of 2). The aspects assessed – information design (ID), navigation design (ND), graphic design (GD), and experience design (ED) – are denoted with a √. Hit and miss accuracies are only reported for discriminating measures.

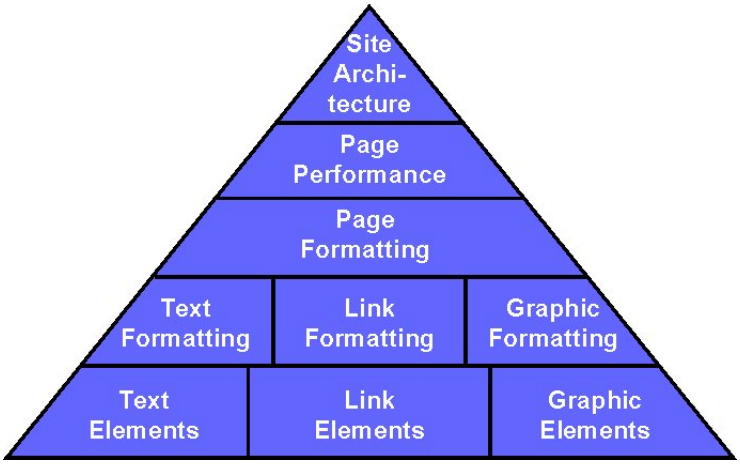


Figure 5.5: Aspects associated with Web interface structure. This is a repetition of Figure 5.2.

Site variation measures are only computed for sites with at least 5 pages of metrics data. Data from 333 sites discussed in Chapter 6 was explored to determine the median CoV for each page-level measure. For each group of measures, metrics that exhibited median Coefficients of Variation smaller than 250% were eliminated, since they could potentially dampen the variation in the remaining measures. This cutoff eliminated nine text element measures, including the Word, Page Title, and Good Body Word Counts discussed in Sections 5.4.1 and 5.4.2. Page depth and type as well as the term measures used for scent quality assessment (e.g., Visible Page Text Terms and All Unique Link Text Terms; Section 5.12.7) were also excluded, since they are intended to be used for reference. Page Title Variation only considers the page title hits and scores; this provides some insight about the use of different page titles throughout the site [Nielsen 2000].

Although the site variation measures are computed with high accuracy, this is not a reflection of how accurately the measures align with users' perceptions. How users gauge site consistency will be verified with a future study. Analysis in Chapter 6 should identify a subset of measures for these studies. Future work will also entail measuring the consistent use of image, script, stylesheet, and other files across pages in a site.

Another limitation of the site variation measures is that all types of pages (home, link, and content) are grouped together in computations. For example, link pages may have a substantially larger number of links than content pages, which could inflate the link element variation. A better approach would be to assess the variation within home, link, content, form, and other pages separately and then use a weighting factor to aggregate across page types. Unfortunately, this computation would require at least five pages of each type, which may not always be possible.

5.13.2 Site Architecture Measures: Size

The size of a site (i.e., the number of pages or documents) is used in the literature for classifying sites into genres, such as newspaper, course outline, and book [Bauer and Scharl 2000; Shneiderman 1997]. In addition, the breadth (how many links are presented on a page) and depth (how many levels must be traversed to find information) of pages within the site has been associated with the quality of the information architecture [Chi *et al.* 2000; Fleming 1998; Furnas 1997; Larson and Czerwinski 1998; Miller and Remington 2000; Nielsen 2000; Rosenfeld and Morville 1998; Sawyer *et al.* 2000; Spool *et al.* 1999; Spool *et al.* 2000]. Several usability studies have been conducted to provide guidance about the breadth and depth of sites. For example, Larson and Czerwinski [1998] suggest that Web designers use moderate levels of breadth with minimal depth (e.g., two levels) in the information architecture. As another example, Rosenfeld and Morville [1998] suggest that Web designers minimize the number of options on the home page to ten and minimize depth to less than five levels.

Several measures were developed to possibly provide some insight into the size, breadth, and depth of a site, including the Page Count, Maximum Page Depth, and Maximum and Median Page Breadths. These measures are based on the configuration of the crawler at the time of data collection as well as the number of pages for which page-level metrics are computed. Hence, they may not accurately reflect actual site characteristics even though they are computed accurately. If the crawler is configured for unlimited crawling and page-level metrics are computed for all of the downloaded pages, then these measures will actually reflect the degree to which the site conformed to crawler restrictions (e.g., pages at subsequent levels were not visible at the previous level, pages are not advertisements, guestbook, or chat room pages, and pages are not pdf, Word, or other documents). Future work will involve capturing more definitive measures of site size.

5.14 Summary

This chapter presented a view of Web interface structure and 157 highly-accurate, page-level and site-level measures for quantifying many aspects, including the amount and type of text on a page, page download speed, colors and fonts used on the page, similarity in content between pairs of pages, and the amount of variation in measures across pages in a site. Quantifying Web interface aspects makes it possible to develop statistical models for distinguishing good interfaces; this is the subject of the next chapter. As shown in this chapter, design guidance can sometimes be contradictory; quantifying Web interface aspects also makes it possible to provide concrete design guidance.

Subsequent chapters use quantitative measures computed for a large collection of expert-rated Web sites to develop several statistical models for assessing Web interface quality. Chapter 7 describes a study on linking the profiles to usability. Chapter 8 demonstrates the use of profiles to assess an example site, and Chapter 9 describes a study that examines the efficacy of the profiles in improving the design of the example site and four others. Finally, Chapter 10 demonstrates the use of statistical models to examine many Web design guidelines presented in this chapter.

Chapter 6

Profiles of Highly-Rated Web Interfaces

6.1 Introduction

The focus of this chapter is the development of statistical models or profiles to support assessing Web site quality. Although statistical model development is a common methodology used for solving many problems from evaluating credit card applicants to personalizing information displayed to Web site visitors, this approach has not been previously used for evaluating Web sites. The use of extensive quantitative measures (described in Chapter 5) combined with Internet professionals' ratings for a large collection of Web sites makes it possible to apply model development techniques towards the problem of automated analysis of Web interfaces.

This chapter begins with a brief discussion of two prior studies that demonstrated the feasibility of developing statistical models to predict interface quality. Then, it describes the most recent study that shows that sophisticated statistical models can be developed to predict interface quality at both the page and site levels while taking into consideration the type of content on a site and the functional style of a page, for example. A shorter version of the study is scheduled for publication [Ivory and Hearst 2002].

6.2 Background: Prior Profile Development Work

Two prior studies by this author established that statistical models could be developed to predict interface quality from quantitative Web interface measures and corresponding expert ratings [Ivory *et al.* 2000; Ivory *et al.* 2001]. The first study reported a preliminary analysis of a collection of 428 Web pages [Ivory *et al.* 2000]. Each page corresponded to a site that had either been rated by Internet experts or had no rating. The expertise ratings were derived from a variety of sources, such as *PC Magazine's* Top 100, WiseCat's Top 100, and the final nominees for the 1999 Webby Awards; if a site was acknowledged by one of these sources, then it was considered to be rated. For each Web page, twelve quantitative measures having to do with page composition, layout, amount of information, and size (e.g., number of words, links, and colors) were computed.

Results showed that six measures – text cluster count, link count, page size, graphics count, color count, and reading complexity – were significantly associated with rated sites. Additionally, two strong pairwise correlations for rated sites, and five pairwise correlations for unrated sites were revealed. Predictions about how the pairwise correlations were manifested in the layout of the rated and unrated sites' pages were supported by inspection of randomly selected pages. A linear

discriminant classifier applied to the groups (rated versus unrated) was able to classify pages into the two groups with 63% accuracy. The study also showed that the predictive accuracy could be improved by considering the functional type – home or other – in models.

The second study reported an analysis of 1,898 pages from 163 sites evaluated for the Webby Awards 2000 [The International Academy of Arts and Sciences 2000; Ivory *et al.* 2001]. At least three Internet professionals (referred to as expert judges) evaluated sites on six criteria: content, structure and navigation, visual design, functionality, interactivity, and overall experience; the six criteria were highly correlated and were summarized with one factor derived via principal components analysis [Sinha *et al.* 2001]. Pages were from sites in six topical categories – community, education, finance, health, living, and services – and represented several groups of sites, as rated by judges: good (top 33% of sites); "not good" (remaining 67% of sites), and poor (bottom 33% of sites). All of the quantitative measures examined in the first study were used, except for reading complexity. The reading complexity measure – the Gunning Fog Index [Gunning 1973] – was not used in this study because it was not computed for many small pages; the index requires at least a hundred words on a page for computation.

The analysis methodology of the first study was replicated to develop two linear discriminant classifier models: 1. distinguishing pages from good and not good sites; and 2. distinguishing pages from good and poor sites. For the first model, the predictive accuracy was 67% when content categories (e.g., community and education) were not taken into account and ranged between 70.7% and 77% when categories were assessed separately. The predictive accuracy of the second model ranged between 76% and 83%. Analysis of individual measures revealed that the word count could be used to characterize sub-groups of good pages. For example, good pages with a low word count (66 words on average as compared to 230 and 827 words for medium and large pages, respectively) had slightly more content, smaller page sizes, less graphics, and used more font variations than corresponding not-good pages.

6.3 Data Collection

The analysis in this chapter uses a large collection of pages and sites from the Webby Awards 2000 dataset [The International Academy of Arts and Sciences 2000] and is similar to the second study [Ivory *et al.* 2001]. This dataset as well as the analysis data are described below.

6.3.1 The Webby Awards 2000

The Webby Awards dataset is a unique untapped resource, as it appears to be the largest collection of Web sites rated along one set of criteria. For the 2000 awards, an initial pool of 2,909 sites were rated on overall quality as well as five specific criteria: content, structure & navigation, visual design, functionality, and interactivity. Additionally, the Web sites were assigned into 27 content categories, including news, personal, finance, services, sports, fashion, technology, arts, and weird. A panel of over 100 judges from The International Academy of Digital Arts & Sciences used a rigorous evaluation process to select winning sites. Webby Awards organizers state the judge selection criteria as follows:

> "Site Reviewers are Internet professionals who work with and on the Internet. They have clearly demonstrable familiarity with the category in which they review and have been individually required to produce evidence of such expertise. The site reviewers are given different sites in their category for review and they are all prohibited from

reviewing any site with which they have any personal or professional affiliation. The Academy regularly inspects the work of each reviewer for fairness and accuracy."

The judging takes place in three stages: review, nominating, and final; only the list of nominees for the final round are available to the public. Anyone can nominate any site to the review stage; nearly 3,000 Web sites were nominated in 2000. The analysis in this chapter focuses solely on sites evaluated during the review stage. Sinha *et al.* [2001] conducted an in depth analysis of judges' ratings for this stage and found that the content criterion was the best predictor of the overall score, while visual design was a weak predictor at best. However, all of the criteria are highly correlated and can be summarized with a single factor derived via principles component analysis [SPSS Inc. 1999]; this factor (referred to as the Webby factor) explained 91% of the variance in the criteria.

For the current study, sites were selected from six content categories – community, education, finance, health, living, and services – as described below.

Community. "Sites developed to facilitate and create community, connectedness and/or communication. These sites can target either a broad-based or niche audience."

Education. "Sites that are educational, promote education, or provide online curriculum. This could include educational content for children or adults, resources for educators, and 'distance learning' courses."

Finance. "Sites relating to financial services and/or information. These include online stock trading, financial news, or investor services."

Health. "Sites designed to provide information and resources to improve personal health. These may include medical news sites, health information, and online diagnosis. Health includes not only medicine but also includes alternative and mental health or fitness Web sites."

Living. "Sites which provide content about how to go about daily life or about elements that touch the personal side of life. Living includes gardening, home improvement, interior design, architecture, food, parenting, and similar subjects."

Services. "Sites that allow real world activities to be done online. These include sites that help people find jobs, houses, dates, or which otherwise facilitate offline activities from the keyboard."

These categories were selected because they were both information-centric (a primary goal is to convey information about some topic) and contained at least 100 sites. Although some sites in the financial and services categories had some functional aspects, such as looking up a stock chart or submitting a resume, most of the pages on these sites provided information. Three groups – good (top 33% of sites), average (middle 34% of sites), and poor (bottom 33% of sites) – were defined for analysis based on the overall score. Table 6.1 depicts the overall score used for defining the three classes of pages and sites. It is assumed that ratings not only apply to the site as a whole, but also to individual pages within the site.

6.3.2 Analysis Data

An early version of the Site Crawler Tool was used to download pages from 1,002 sites in the finance, education, community, health, services, and living categories. The crawler was

	Overall	Community	Education	Finance
Good	6.97	6.58	6.47	6.6
Poor	5.47	5.66	5.38	5.8
	Health	Living	Services	
Good	7.9	6.66	7.54	
Poor	6.4	5.66	5.9	

Table 6.1: Overall scores used to classify pages and sites as good (top 33%), average (middle 34%), or poor (bottom 33%). The rating scale is from one to ten. Average pages and sites fall in the range between the Top and Bottom cutoffs.

configured to crawl two levels from the home page on each site and to download fifteen level-one pages and 45 level-two pages (three pages linked from each level-one page). This number of pages was not retrievable on all of the sites. The early crawler version used for this data set did not follow redirects or download scripts, object files, or images used as form buttons. Hence, measures associated with these elements (e.g., script, object, and graphic bytes; Section 5.12) may be underestimated somewhat.

The Metrics Computation Tool was then used to compute page-level and site-level measures for downloaded pages that contained at least 30 words and were in English. The 30-word limit was used to eliminate blank or error message pages but still retain smaller content or form pages. The final data collection consists of 5,346 pages from 639 sites. The collection includes data for good, average, and poor pages and sites within each of the six content categories. Four of the good pages have missing Bobby measures; hence, some techniques, such as discriminant classification, exclude these pages, while other techniques, such as multiple linear regression and decision tree modeling, do not. Only 333 of the 639 sites have at least five downloaded pages as required for computing the variation measures (see Section 5.13.1); the site-level analysis will consider only this subset. The small number of sites with at least five pages can be attributed to several restrictions on the Site Crawler Tool, including a 12-minute crawling time limit on sites and the requirement that pages at subsequent crawling levels were not previously accessible.

All of the measures, except the two reading complexity measures, will be used for analyses throughout this chapter; reading complexity is excluded due to a high number of cases wherein reading complexity could not be computed[1]. However, individual measures used to derive reading complexity (e.g., fog big word count and fog sentence count; Section 5.4.11) are included in the analyses. All of the measures were carefully screened to remove outliers within the three classes of pages; the resulting data was normally distributed with equal variances.

6.4 Profile Development Methodology

The goal of profile development is to derive statistical models for classifying Web pages and sites into the good, average, and poor classes based on their quantitative measures. As discussed in Chapter 4, profile development encompasses statistical analysis of quantitative measures (page-level and site-level) and corresponding expert ratings. Prior studies included univariate and multivariate analyses [Ivory *et al.* 2000; Ivory *et al.* 2001]. Statistical techniques, such as correlation coefficients and t-tests for equality of means [Keppel and Zedeck 1989], revealed significant relationships among individual measures and expert ratings within each class of pages (i.e., rated vs. unrated and

[1]The Gunning Fog Index requires 100 words for computation; hence, it may not be possible to compute this index for pages with small amounts of text.

highly-rated vs. poorly-rated). Two multivariate techniques – multiple linear regression and linear discriminate analysis – illustrated significant relationships between measures as well as key measures for predicting the class of each page.

The prior analyses focused on describing key differences between the classes of pages; the analysis in this chapter expands on this work and also explores properties of highly-rated pages. The three-step analysis approach is described below.

1. Develop a model to classify pages into the three classes – good, average, and poor. Use linear discriminant classification and decision tree modeling as necessary.

2. For pages accurately classified by the model as good, identify groups of pages with common properties. Use K-means clustering [SPSS Inc. 1999] in this step.

3. Examine key relationships among measures in each cluster of good pages. Use descriptive statistics, ANOVAs, and correlation coefficients as necessary.

Comparisons are also made among good, average, and poor pages using techniques similar to step 3 above. The first and third steps are also followed for site-level analysis. This chapter replicates analyses performed in prior studies with several key differences: a larger sample of pages is used; site-level analysis is included; three classes of pages and sites are contrasted (good, average, and poor versus top and bottom); a larger number of quantitative measures are used; a page type is incorporated into the analysis; and machine learning techniques are used to develop models in some cases.

Profile development is approached in several phases in this chapter. First, profiles are developed across all of the pages irrespective of page types and content categories. Then, page types (home, link, content, form, and other; see Section 5.11) and content categories (community, education, finance, health, living, and services) are considered separately. Finally, profiles are developed across sites and within content categories across sites.

The profile development work revealed interesting correlations for measures within the good, average, and poor classes; however, it is not suggested that these correlations caused ratings. Causality can only be established with controlled usability studies; this will be the focus of future work. The study of pages and sites modified based on the developed profiles does suggest that some design aspects gleaned from the profiles are viewed favorably by users; Chapter 9 discusses this study.

6.5 Summary of Developed Profiles

Several statistical models were developed to classify pages and sites into the three classes (good, average, and poor) as depicted in Table 6.2. Another model was developed to map pages into one of the three clusters of good pages: small-page, large-page, and formatted-page. Each of these models encapsulates key predictor measures and relationships among measures for classifying pages and sites. All of the models are used by the Analysis Tool (see Chapters 4 and 8) and are summarized below. The remainder of this chapter discusses each model in detail.

6.5.1 Page-Level Models

Overall Page Quality (Section 6.6): a decision tree model for classifying a page into the good, average, and poor classes without considering the functional type of a page or the content category (see below). The model also reports the decision tree rule that generated the prediction.

Assessment Type	Analysis Method	Accuracy		
		Good	Average	Poor
Page Level (5346 pages)				
Overall Quality	C&RT	96%	94%	93%
Page Type Quality	LDA	84%	78%	84%
Content Category Quality	LDA	92%	91%	94%
Site Level (333 sites)				
Overall Quality	C&RT	88%	83%	68%
Content Category Quality	C&RT	71%	79%	64%

Table 6.2: Page and site level classification accuracies. C&RT refers to the Classification and Regression Tree algorithm. LDA refers to the Linear Discriminant Analysis.

Closest Good Page Cluster (Section 6.7): a K-means clustering model for mapping a page into one of the three good page clusters – small-page, large-page, and formatted-page. The model reports the distance between a page and the closest cluster's centroid and the top ten measures that are consistent with this cluster. The model also reports the top 10 measures that are inconsistent with the cluster as well as acceptable metric ranges. In both cases, measures are ordered by their importance in distinguishing pages in the three clusters as determined from ANOVAs.

Page Type Quality (Section 6.8): discriminant classification models for classifying a page into the good, average, and poor classes when considering the functional type of a page – home, link, content, form, and other. The model reports the top 10 measures that are consistent with the page type. The model also reports the top ten measures that are inconsistent with the page type and acceptable metric values. In both cases, measures are ordered by their importance in distinguishing pages in the good, average, and poor classes as determined from ANOVAs. A separate decision tree model predicts the functional type of a page based on page-level measures (see Section 5.11). The Analysis Tool also enables users to specify a page type for analysis.

Content Category Quality (Section 6.9): discriminant classification models for classifying a page into the good, average, and poor classes when considering the content category of the site – community, education, finance, health, living, and services. Each model reports the top ten measures that are consistent with the content category. Each model also reports the top ten measures that are inconsistent with the content category and acceptable metric values. In both cases, measures are ordered by their importance in distinguishing pages in the good, average, and poor classes as determined by ANOVAs. The Analysis Tool enables users to specify content categories for analysis.

An effort was made to develop good page clusters for each of the content categories and page types, but the number of good pages in each category was inadequate. K-means clustering typically converged onto two or three clusters; however, the cluster sizes were disproportionate with one or two clusters containing less than 30 pages, for instance.

Similarly, an effort was made to develop classification models that consider page type and content category combinations (i.e., predicting good community home pages or good finance link pages). Table 6.3 shows the distribution of pages into each content category and page type combination, and Table 6.4 shows the accuracy of discriminate classification models developed

Cont. Cat.	Page Type														
	Home			Link			Content			Form			Other		
	G	A	P	G	A	P	G	A	P	G	A	P	G	A	P
Comm.	67	33	30	190	108	51	212	155	101	51	53	19	22	16	15
Educ.	53	53	35	189	145	119	184	163	171	82	47	29	11	25	37
Finance	24	13	25	67	60	64	107	47	104	30	5	38	5	2	10
Health	20	33	26	61	145	90	67	218	153	21	46	21	3	15	9
Living	36	23	20	113	53	90	80	83	84	30	31	20	8	21	8
Services	15	32	23	60	70	65	71	86	108	20	35	19	5	19	21

Table 6.3: Number of good (G), average (A), and poor (P) pages used to develop models for each content category and page type combination. Four pages with missing Bobby measures were discarded by the discriminant classification algorithm.

for each combination. In some cases predictive accuracy is considerably lower, possibly due to inadequate data. It is also possible that page type mispredictions contribute to lower predictive accuracy. However, the results suggest that these models could be built with at least 60 pages for each combination.

6.5.2 Page-Level Models: Key Predictor Measures

Tables 6.5, 6.6, and 6.7 summarize page-level measures that were among the top ten measures in the models for classifying good, average, and poor pages (described above); ANOVAs were used to determine the top ten predictor measures for each model. The tables show that several measures – italicized body word count (text formatting), minimum font size (text formatting), minimum color use (page formatting), and Weblint errors (page performance) – were among the top ten predictors in over half of the models. The analysis showed that pages with many italicized body text words were consistent with poor pages. The analysis also showed that good pages use a smaller font size (< 9 pt), typically for copyright text, and an accent (color sparsely used; measured by the minimum color use). Finally, the analysis showed that good pages tended to contain more HTML coding (Weblint) errors, which correlated with page formatting measures, such as the table and interactive object counts.

Tables 6.5, 6.6, and 6.7 also show that the other predictor measures vary considerably across the page-level models. All but three models – other, health, and living page quality – have one or more text element measures as key predictors. Only a few models (overall, home, health, and living page quality) have link element measures as key predictors; the health page quality model has four link element measures as key predictors. The graphic element measures are used more so for the overall and page type quality models than for the content category quality models. The text formatting, graphic formatting, page formatting, and page performance measures are used fairly equally among the models. The link formatting measures are used more so for the content category quality models than for the overall and page type quality models.

The variation among key predictors in the models suggests that characteristics of pages vary depending upon the context – page type or content category. Consequently, design goals need to be clarified before applying the models towards assessing and improving a Web interface. Variation in key predictor measures plus the fact that it was possible to get accurate predictions from a small number of pages in the combined page type and content category models, suggest that an extensive set of page-level measures, such as the ones developed, is essential to model

Cont. Cat.	Page Type								
	Home			Link			Content		
	G	A	P	G	A	P	G	A	P
Comm.	79%	64%	83%	99%	100%	96%	97%	99%	92%
Educ.	94%	89%	94%	96%	90%	95%	94%	96%	92%
Finance	75%	85%	84%	100%	100%	100%	100%	100%	100%
Health	75%	64%	89%	98%	97%	98%	99%	99%	98%
Living	61%	70%	55%	93%	94%	92%	100%	98%	99%
Services	60%	69%	65%	100%	100%	100%	100%	100%	100%
Cont. Cat. Avg.	74%	73%	78%	98%	97%	97%	98%	98%	98%

Cont. Cat.	Page Type					
	Form			Other		
	G	A	P	G	A	P
Comm.	92%	93%	74%	73%	88%	80%
Educ.	83%	83%	90%	100%	88%	89%
Finance	100%	100%	95%	100%	100%	90%
Health	100%	98%	95%	100%	53%	100%
Living	100%	94%	85%	75%	95%	100%
Services	90%	100%	100%	100%	79%	95%
Cont. Cat. Avg.	94%	95%	90%	91%	84%	92%

Table 6.4: Classification accuracy for predicting good (G), average (A), and poor (P) pages for each content category and page type combination.

Measure	Overall Quality	Page Type Qual.					Content Cat. Qual.						Use Freq.
		H	L	C	F	O	C	E	F	H	L	S	
Text Element Measures													
Link Word Count		√	√										16.67%
Good Link Word Count	√		√										16.67%
Graphic Word Count					√			√					16.67%
Good Graphic Word Count								√	√				16.67% 16.67%
Spelling Error Count				√			√					√	25.00%
Link Element Measures													
Text Link Count	√												8.33%
Link Count		√								√			16.67%
Internal Link Count		√								√	√		25.00%
Redundant Link Count										√			8.33%
Link Graphic Count										√			8.33%
Graphic Element Measures													
Graphic Ad Count	√	√	√		√								33.33%
Animated Graphic Ad Count			√				√				√		25.00%
Text Formatting Measures													
Italicized Body Word Count	√	√		√	√	√		√		√			58.33%
Exclaimed Body Word Count		√											8.33%
Bolded Body Word Count						√							8.33%
Minimum Font Size	√		√	√	√		√	√			√		58.33%
Body Color Count					√								8.33%
Text Cluster Count			√										8.33%
Link Text Cluster Count												√	8.33%
Text Column Count			√							√		√	25.00%

Table 6.5: Key measures used for predictions in the page-level models; all of these measures were among the top 10 predictors for at least one model (Table 1 of 3). The page type quality models are for home (H), link (L), content (C), form (F), and other (O) pages. The content category models are for pages from community (C), education (E), finance (F), health (H), living (L), and services (S) sites. A √ indicates whether a measure was among the top 10 predictors for a model. The use frequency reflects the percentage of time a measure is among the top predictors across all of the models.

Measure	Overall Quality	Page Type Qual.					Content Cat. Qual.						Use Freq.
		H	L	C	F	O	C	E	F	H	L	S	
Link Formatting Measures													
Link Color Count											√		8.33%
Standard Link Color Count					√						√		16.67%
Non-Underlined Text Links							√		√			√	25.00%
Graphic Formatting Measures													
Minimum Graphic Width				√	√	√					√		33.33%
Minimum Graphic Height											√		8.33%
Page Formatting Measures													
Minimum Color Use	√		√	√		√	√	√				√	58.33%
Interactive Object Count					√			√					16.67%
Search Object Count									√		√	√	25.00%
Good Text Color Combinations									√				8.33%
Neutral Text Color Combinations						√							8.33%
Good Panel Color Combinations				√									8.33%
Bad Panel Color Combinations		√											8.33%
Vertical Scrolls				√									8.33%
Horizontal Scrolls				√									8.33%
Serif Font Count									√				8.33%
Undetermined Font Style Count							√						8.33%
Fixed Page Width Use								√					8.33%
Internal Stylesheet Use									√				8.33%
Self Containment											√		8.33%

Table 6.6: Key measures used for predictions in the page-level models; all of these measures were among the top 10 predictors for at least one model (Table 2 of 3). The page type quality models are for home (H), link (L), content (C), form (F), and other (O) pages. The content category models are for pages from community (C), education (E), finance (F), health (H), living (L), and services (S) sites. A √ indicates whether a measure was among the top 10 predictors for a model. The use frequency reflects the percentage of time a measure is among the top predictors across all of the models.

Measure	Overall Quality	Page Type Qual.					Content Cat. Qual.						Use Freq.
		H	L	C	F	O	C	E	F	H	L	S	
Page Performance Measures													
Graphic File Count												√	8.33%
HTML File Count									√				8.33%
Script File Count							√						8.33%
Script Bytes									√				8.33%
Object Count		√					√	√		√			33.33%
Table Count												√	8.33%
Bobby Approved												√	8.33%
Bobby Priority 1 Errors										√			8.33%
Bobby Priority 2 Errors	√	√						√					25.00%
Bobby Browser Errors				√	√			√		√			33.33%
Weblint Errors	√	√	√		√					√	√	√	58.33%
Visible Page Text Hits						√							8.33%
All Page Text Hits						√							8.33%
Visible Link Text Hits						√							8.33%
Visible Link Text Score						√							8.33%
All Link Text Hits						√							8.33%
Page Title Hits							√		√				16.67%
Page Title Score				√			√		√				25.00%

Table 6.7: Key measures used for predictions in the page-level models; all of these measures were among the top 10 predictors for at least one model (Table 3 of 3). The page type quality models are for home (H), link (L), content (C), form (F), and other (O) pages. The content category models are for pages from community (C), education (E), finance (F), health (H), living (L), and services (S) sites. A √ indicates whether a measure was among the top 10 predictors for a model. The use frequency reflects the percentage of time a measure is among the top predictors across all of the models.

development. Both prior metric studies used only a small subset of measures and a larger number of pages than the combined models, but the predictive accuracy was considerably less. Hence, it appears that the high accuracy of the developed models is largely attributable to the exhaustive set of measures.

6.5.3 Site-Level Models

One limitation of the site-level models below is that they do not take page-level quality into consideration. Thus, it is possible for a site to be classified as good even though all of the pages in the site are classified as poor and vice versa. To remedy this situation, the median predictions for pages in the site are also reported by the Analysis Tool. These median page-level predictions need to be considered in determining the overall quality of a site.

Overall Site Quality (Section 6.10): a decision tree model for classifying a site into the good, average, and poor classes without considering the content category (see below). The model also reports the decision tree rule that generated the prediction.

Median Overall Page Quality (Section 6.10): predictions from the overall page quality model (described in Section 6.6) are used to derive the median overall page quality; the median overall page quality is then used to classify a site into the good, average, and poor classes. These predictions need to be considered in conjunction with predictions from the overall site quality model above. The accuracy of this model is the same as the accuracy of the overall page quality model; hence, no accuracy measure is reported in Table 6.2.

Content Category Quality (Section 6.11): decision tree models for classifying a site into the good, average, and poor classes when considering the content category of the site – community, education, finance, health, living, and services. Each model reports the decision tree rule that generated the prediction.

Median Content Category Quality (Section 6.11): predictions from the page-level content category quality models (described in Section 6.9) are used to derive the median content category quality; the median content category quality is then used to classify a site into the good, average, and poor classes. These predictions need to be considered in conjunction with predictions from the site-level content category quality models above. The accuracy of these models are the same as the accuracy of the content category models for pages; hence, no accuracy measure is reported in Table 6.2.

To evaluate sites with the site-level models, the Site Crawler Tool (see Chapter 4) needs to be used to download pages from sites. The crawler should be configured to crawl three levels on each site and to download fifteen level-one pages and three level-two pages linked from each level-one page.

6.5.4 Site-Level Models: Key Predictor Measures

The maximum page depth was the only significant measure in site quality predictions, specifically for the overall site quality model; this is possibly due to inadequate data, the need for better site measures, or the need for model building methods. Table 6.2 shows that the site-level models are considerably less accurate than the page-level models. Future work will explore better measures and possibly model building methods to improve the site-level models.

Function	Squared Canonical Correlation	Wilks' Lambda	Chi-Square	Sig.	Classification Accuracy		
					Good	Average	Poor
1	0.44	0.394	4915.8	0.000	–	–	–
2	0.29	0.709	1810.5	0.000	–	–	–
Overall	–	–	–	–	76%	67%	74%

Table 6.8: Classification accuracy for predicting good, average, and poor pages using two linear discriminant functions.

6.6 Overall Page Quality

The goal of this section is to present an overall view of highly-rated pages that can be used in the future to assess pages without considering the six content categories studied. This analysis also does not consider page types. The data consists of 5,346 pages – 1,906 good pages (36%), 1,835 average pages (34%), and 1,605 poor pages (30%).

6.6.1 Overall Page Quality: Classification Model

Linear discriminant classification was used to develop a model for classifying all of the pages into the good, average, and poor classes. All of the measures, except external stylesheet use, met the criteria for inclusion in model development[2]. Table 6.8 summarizes key classification accuracy measures for this model. Since classification is performed for three groups, the algorithm derived 2 classification functions. The squared canonical correlation indicates the percentage of variance in the measures accounted for by each discriminant function. Wilks' Lambda indicates a complementary measure – the proportion of variance not explained by differences among groups. The corresponding Chi-Square for each Wilks' Lambda is computed for reporting significance; both discriminant functions have significant Wilks' Lambda. The model classifies pages with 72% accuracy overall.

Standardized coefficients for both discriminant functions illustrate key measures for classifying pages. Measures with standardized coefficients greater than one (in absolute value) are listed below.

Function 1: text element measures – meta tag, good meta tag, page title, and overall page title word counts; and page performance measures – page title and unique page title terms.

Function 2: text element measures – good page title, and overall good page title word counts, and fog sentence count; graphic element measure – graphic count; and page performance measures – visible page, visible link, all link, and all unique link text terms, page title and unique page title terms.

Classification accuracy was improved by developing a decision tree with the Classification and Regression Tree (C&RT) algorithm [Breiman *et al.* 1984]; 70% of the data was used for training and 30% for the test sample. The resulting tree contains 144 rules and has an overall accuracy of 94% (96%, 94%, and 93% for good, average, and poor pages, respectively). The tree uses 71 of the measures; these measures represent all eight of the page-level metric categories – text element and formatting, link element and formatting, graphic element and formatting, and page formatting

[2]The page depth, reading complexity, and overall reading complexity measures were excluded from analyses in this section and others. Page type is also excluded except in sections examining this measure.

if ((Italicized Body Word Count is missing OR (Italicized Body Word Count $\leq$ 2.5)) AND (Minimum Font Size is missing OR (Minimum Font Size $\leq$ 9.5)) AND (Graphic Ad Count is not missing AND (Graphic Ad Count $>$ 2.5)))
Class = Good

This rule classifies pages as good pages if they have: two or fewer italicized body text words; use a font size of 9pt or less for some text; and more than two graphical ads.

---

if ((Italicized Body Word Count is missing OR (Italicized Body Word Count $\leq$ 2.5)) AND (Minimum Font Size is missing OR (Minimum Font Size $\leq$ 9.5)) AND (Graphic Ad Count is missing OR (Graphic Ad Count $\leq$ 2.5)) AND (Exclaimed Body Word Count is missing OR (Exclaimed Body Word Count $\leq$ 12.5)) AND (Exclaimed Body Word Count is not missing AND (Exclaimed Body Word Count $>$ 11.5)) AND (Bobby Priority 2 Errors is missing OR (Bobby Priority 2 Errors $\leq$ 5.5)) AND (Meta Tag Word Count is missing OR (Meta Tag Word Count $\leq$ 66)) AND (Emphasized Body Word Count is missing OR (Emphasized Body Word Count $\leq$ 174.5)) AND (Bad Panel Color Combinations is missing OR (Bad Panel Color Combinations $\leq$ 2.5)))
Class = Average

This rule classifies pages as average pages if they have: two or fewer italicized body text words; use a minimum font size of 9pt or less for some text; two or fewer graphical ads; twelve exclaimed body words (i.e., body text followed by exclamation points); five or fewer Bobby priority 2 errors; 66 or fewer meta tag words; 174 or fewer emphasized body words (i.e., body text that is colored, bolded, italicized, etc.); and less than two bad panel color combinations.

---

if ((Italicized Body Word Count is not missing AND (Italicized Body Word Count $>$ 2.5)))
Class = Poor

This rule classifies pages as poor pages if they have more than two italicized body text words.

Figure 6.1: Example decision tree rules for predicting page classes (Good, Average, and Poor).

and performance measures. Figure 6.1 depicts example rules for classifying pages into the good, average, and poor classes. Model predictions were retained for further analysis.

6.6.2 Overall Page Quality: Characteristics of Good, Average, and Poor Pages

Further analysis was conducted to determine significant differences between pages in the three classes. Specifically, one-way analyses of variance (ANOVAs) were computed to identify measures where the within-class variance is significantly different from the between-class variance. Correlation coefficients were also computed between pairs of predictor measures. The analysis only considered pages accurately classified by the decision tree – 1,822 good pages, 1,732 average pages, and 1,486 poor pages.

ANOVAs revealed that all but eight of the 71 measures – unique page title terms, graphic count, average font size, maximum graphic height, graphic link count, link graphic count, download time, and whether a page was Bobby approved – were significantly different between the three

classes of pages. Tables 6.9, 6.10, and 6.11 depict means and standard deviations for each measure. The contribution of each measure is reported by the F value; F values were sorted to determine a measure's rank. All of the F values are significant at the .05 level.

The top ten predictors are minimum font size, minimum color use, italicized body word count, Weblint errors, graphic ad count, link text cluster count, interactive object count, Bobby priority 2 errors, text link count, and good link word count. Differences among good, average, and poor pages are described below. ANOVAs were also computed between pairs of classes (i.e., good vs. average, good vs. poor, and average vs. poor) to gain more insight about similarities and differences among the classes; all differences were significant, except as noted below.

- Good pages surprisingly use minimum font sizes of nine points; however, the standard deviation is smaller than those for the other two classes indicating less variance. Inspection of a random sample of good pages revealed that this minimum font size is often used for footer text, such as copyright notices. There is no significant difference between the minimum font sizes employed on average and poor pages.

- Average and poor pages have larger minimum color usages than good pages (five and six times vs. four times). Inspection of a random sample of good average, and poor pages suggest that this results from a tendency for good pages to have at least one sparsely used accent color.

- Good and average pages rarely contain italicized words within body text; there is no significant difference between these two classes. Poor pages contain one italicized body word on average.

- Good pages contain the most Bobby priority 2 and Weblint errors (average of 35 and 19, respectively), while poor pages contain the fewest errors. There were correlations between these errors and the number of interactive objects, tables, images, etc. This finding suggests that highly-rated pages tend not to conform to accessibility and good HTML coding standards. It is possible that in some cases good pages (and possibly average and poor pages) are unnecessarily penalized by these tools. As an example, Bobby requires alternative text to be provided for all images on a page. However, designers may frequently use blank images as spacers and may not provide alternative text for them, resulting in the page not being Bobby approved. On the other hand, if designers did provide alternative text for spacer images, this may actually impede blind users, since the text will be read by screen readers. Perhaps these tools need to consider the context in which page elements are being used.

- Good pages typically contain one graphical ad; poor pages are slightly more likely to contain graphical ads than average pages. An examination of ten sites suggests that ads on good sites are for well-known entities whereas ads on poor sites are for obscure entities. Kim and Fogg [1999] conducted a controlled study wherein 38 users rated Web pages (with and without graphical ads) on credibility ("high level of perceived trustworthiness and expertise") and found that pages with graphical ads were rated as more credible than those without graphical ads.

- Good pages contain about 27 text links, while average pages contain 22 and poor pages contain 19. Poor pages are also less likely to contain link text clusters (areas containing text links highlighted with color, lists, etc.), while good pages contain slightly more link clusters than average pages. There is a corresponding higher number of good link words (words in the link text that are not stop words or 'click') on good and average pages than on poor pages.

Measure	Mean			Std. Dev.			F val.	Rank
	Good	Avg.	Poor	Good	Avg.	Poor		
Text Element Measures								
Good Link Word Count	47.5	36.2	31.8	42.2	34.5	31.0	83.8	10
Good Meta Tag Word Count	15.3	10.9	17.0	23.5	17.2	24.5	33.8	23
Meta Tag Word Count	20.8	14.3	21.6	32.1	22.8	31.3	31.6	24
Good Word Count	204.0	175.4	185.5	157.9	142.9	144.2	16.9	32
Word Count	378.0	326.2	345.5	298.3	271.1	273.3	15.4	37
Good Page Title Word Count	3.1	3.3	3.4	1.7	1.7	1.9	10.9	44
Exclamation Point Count	1.2	1.0	1.0	1.5	1.4	1.4	9.8	45
Display Word Count	17.1	14.8	16.4	17.5	16.0	17.0	8.2	52
Fog Big Word Count	33.7	32.2	36.4	34.3	34.6	37.6	5.7	59
Body Word Count	264.5	243.3	265.1	232.6	224.9	238.5	4.9	60
Good Body Word Count	127.3	118.7	129.2	114.9	113.4	118.3	3.9	62
Link Element Measures								
Text Link Count	27.4	21.5	18.6	23.0	19.5	16.9	84.2	9
Link Count	41.2	34.1	31.8	28.8	23.9	21.7	63.5	14
Redundant Link Count	7.7	6.6	6.7	7.7	6.4	6.8	13.7	39
Graphic Element Measures								
Graphic Ad Count	1.2	0.7	0.7	1.4	0.9	0.8	103.7	5
Redundant Graphic Count	9.4	7.9	8.8	12.4	10.2	11.0	8.1	54
Text Formatting Measures								
Minimum Font Size	9.0	9.3	9.3	0.2	0.7	0.7	247.4	1
Italicized Body Word Count	0.5	0.5	1.1	0.9	0.9	1.7	139.3	3
Link Text Count	1.2	1.0	0.6	1.4	1.4	0.8	91.3	6
Text Column Count	4.1	3.4	2.8	3.7	3.0	2.4	76.4	11
Exclaimed Body Word Count	4.1	3.0	2.3	6.4	4.8	3.7	52.1	18
Text Cluster Count	2.2	1.9	1.5	2.3	2.3	1.5	41.2	21
Capitalized Body Word Count	1.7	1.2	1.8	2.5	1.5	2.5	37.7	22
Text Positioning Count	3.5	2.8	3.3	3.3	2.8	3.2	20.9	28
Sans Serif Word Count	227.4	185.7	214.4	239.7	203.4	228.1	15.9	34
Display Color Count	1.5	1.3	1.4	0.9	1.0	0.9	15.5	35
Emphasized Body Word Count	52.8	51.5	61.4	60.8	56.3	69.2	11.8	43

Table 6.9: Means and standard deviations for good, average, and poor pages (Table 1 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution to predictions (Rank column); the rank reflects the size of the F value.

Measure	Mean			Std. Dev.			F val.	Rank
	Good	Avg.	Poor	Good	Avg.	Poor		
Text Formatting Measures								
Bolded Body Word Count	11.4	12.8	14.0	16.1	17.4	20.2	9.1	48
Colored Body Word Count	17.6	20.2	20.6	24.4	26.4	27.4	6.7	55
Serif Word Count	92.4	83.0	83.2	133.2	116.5	124.9	3.2	63
Link Formatting Measures								
Standard Link Color Count	1.1	1.4	1.5	1.2	1.2	1.3	58.8	16
Graphic Formatting Measures								
Minimum Graphic Width	22.9	31.9	20.4	32.5	43.8	28.5	47.1	20
Minimum Graphic Height	10.1	8.7	9.0	13.7	11.1	12.6	6.2	57
Maximum Graphic Width	436.9	456.3	439.5	190.1	175.2	176.0	5.8	58
Page Formatting Measures								
Minimum Color Use	3.5	5.3	6.6	3.1	5.1	6.7	156.2	2
Interactive Object Count	3.2	2.2	1.7	3.7	2.9	2.5	91.1	7
Bad Panel Color Combinations	1.0	0.7	0.6	1.3	0.8	0.8	70.6	12
Vertical Scrolls	2.0	1.6	1.6	1.4	1.0	1.0	60.2	15
Horizontal Scrolls	0.1	0.1	0.0	0.3	0.3	0.2	28.7	25
Color Count	8.0	7.6	7.5	2.7	2.4	2.3	23.0	26
Font Count	5.6	5.3	5.1	2.2	2.5	2.4	22.7	27
Good Text Color Combinations	3.4	3.1	3.1	2.2	1.8	1.7	19.2	29
Page Pixels	802K	731K	747K	475K	439K	445K	11.8	42
Good Panel Color Combinations	0.4	0.5	0.6	0.7	0.7	0.8	8.9	49
Browser-Safe Color Count	5.0	4.9	4.9	1.6	1.6	1.5	4.7	61

Table 6.10: Means and standard deviations for good, average, and poor pages (Table 2 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution to predictions (Rank column); the rank reflects the size of the F value.

Measure	Mean			Std. Dev.			F val.	Rank
	Good	Avg.	Poor	Good	Avg.	Poor		
Page Performance Measures								
Weblint Errors	34.5	26.3	19.1	36.6	27.8	18.4	116.1	4
Bobby Priority 2 Errors	4.0	3.6	3.5	1.3	1.1	1.0	87.3	8
Bobby Browser Errors	13.9	11.5	11.8	7.5	6.3	5.8	66.3	13
Object Count	1.9	1.3	1.4	2.1	1.4	1.4	55.1	17
Script File Count	0.5	0.2	0.2	1.2	0.6	0.6	52.0	19
All Page Text Terms	305.2	265.7	303.3	214.7	191.3	232.5	18.9	30
Visible Page Text Terms	258.7	221.8	254.9	203.4	175.4	212.4	18.3	31
Visible Page Text Hits	31.0	28.5	26.4	24.1	22.9	21.9	16.5	33
Table Count	8.1	7.4	6.9	6.2	5.7	5.7	15.4	36
Script Bytes	1.2K	1.1K	924.0	1.4K	1.5K	1.1K	14.3	38
HTML Bytes	15.4K	14.3K	13.8K	9.5K	9.8K	8.2K	13.0	40
Bobby Priority 1 Errors	1.3	1.2	1.3	1.0	0.8	0.8	13.0	41
All Unique Link Text Terms	133.5	119.4	125.8	101.9	91.2	92.8	9.7	46
All Page Text Score	141.1	126.3	138.2	106.3	101.5	114.2	9.3	47
Visible Page Text Score	91.5	81.0	85.8	77.4	70.8	77.6	8.7	50
Visible Unique Link Text Terms	120.4	108.5	111.4	96.2	87.9	87.6	8.2	51
All Page Text Hits	41.7	37.8	39.2	29.8	28.3	30.2	8.1	53
Page Title Terms	3.8	3.9	4.1	2.4	2.1	2.4	6.6	56

Table 6.11: Means and standard deviations for good, average, and poor pages (Table 3 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution to predictions (Rank column); the rank reflects the size of the F value.

- Good pages appear to be more interactive than pages in the other classes; they contain about three interactive objects (e.g., search button or pulldown menu). Average and poor pages contain about two interactive objects.

Exploring large correlations (i.e., $r \geq .5$ in absolute value) between pairs of measures within each sample provided more insight about differences among the classes as described below.

- Good pages appear to use colors in various ways. Correlation between the color and display color counts suggests that these pages use a multi-level heading scheme wherein headings at each level are different colors. There is also a correlation between good text color and good panel color combinations suggesting these pages use colored areas and colored text simultaneously (e.g., in navigation bars). Good pages also use tables to control the formatting of text links and images. Correlations between redundant link and graphic link counts coupled with a medium-strength correlation between redundant link and text link counts suggest that links are presented multiple times in different forms (e.g., as an image in a navigation bar and as text in a footer).

- Average pages appear to use bad panel color combinations to format link text clusters. Furthermore, the number of good links words is correlated with the number of redundant links suggesting that good link words may appear for example in navigation bars and footers but not necessarily in the text. The average and minimum font sizes are correlated suggesting little variance in text sizes on average pages.

- Poor pages appear to use color to a lesser degree than good and average pages; however, when colors are used, they are typically overused as discussed previously. Color count is correlated with the number of interactive objects suggesting that color is used to highlight these objects. The number of good text color combinations is also correlated with the number of interactive objects, which are typically formatted with a white background and black text. This suggests that poor pages tend to use multiple formatting techniques at once. There is a correlation between redundant graphic count and graphic link count, which suggests that image links are presented multiple times. This is in contrast to good pages that repeat links in multiple forms.

Figures 6.2, 6.3, and 6.4 depict example good, average, and poor pages, respectively. These pages demonstrate many of the discussed properties.

6.7 Good Page Clusters

The decision tree model presented in Section 6.6 accurately classified 1,822 (96%) of the 1,906 good pages from both the training and test sets; these pages were retained for cluster analysis. K-means clustering [SPSS Inc. 1999] was used to identify sub-groups of similar good pages. This method requires all measures to be on the same scale; hence, measures were transformed into Z scores. Each Z score is a standard deviation unit that indicates the relative position of each value within the distribution (i.e., $Z_i = \frac{x_i - \bar{x}}{\sigma}$, where x_i is the original value, $\bar{x}$ is the mean, and σ is the standard deviation).

K-means clustering converged onto three clusters of good pages. The first cluster consists of 450 pages (24.5%), the second cluster consists of 364 pages (20%), and the final cluster consists of 1,008 pages (55.3%). Tables 6.12, 6.13, and 6.14 contrast means and standard deviations for the three clusters and depict the rank of each measure based on ANOVA results(i.e., F values). All

Figure 6.2: Good page exhibiting several key properties of this class: links repeated in multiple forms (text and images), graphical ads, interactive objects, multi-level colored headings, navigation bars, and variations in font sizes, including a smaller size for the footer.

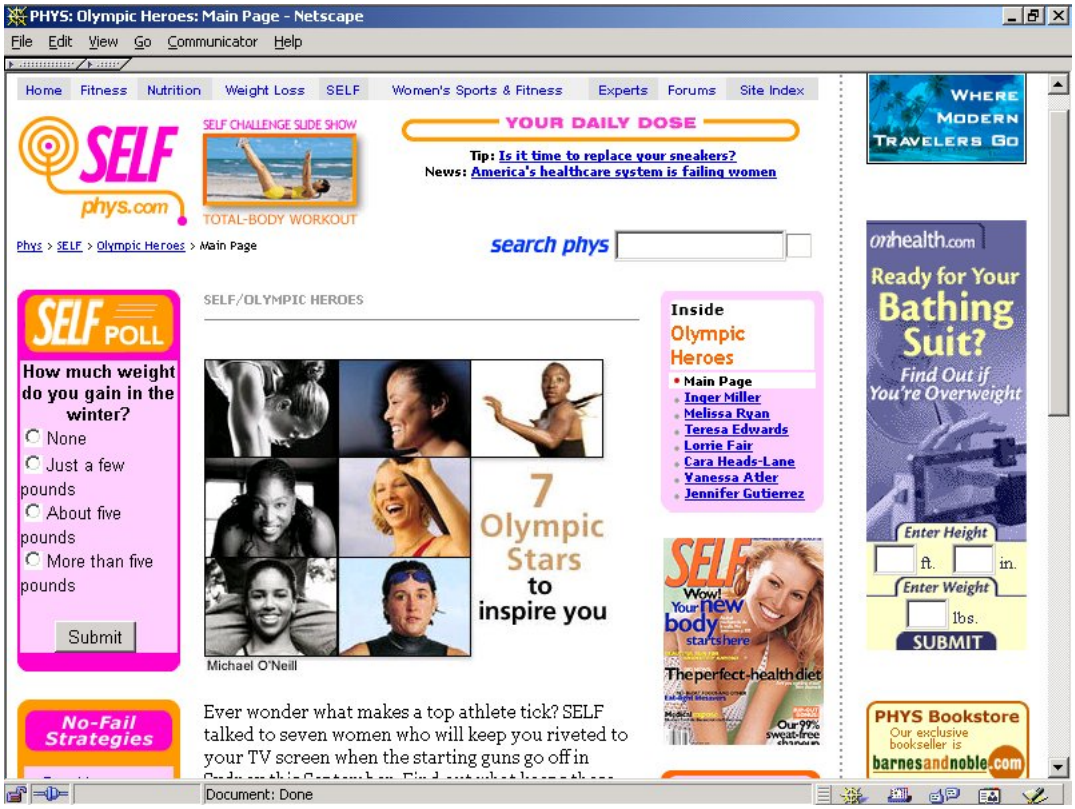


Figure 6.3: Average page exhibiting several key properties of the class: bad panel and text color combinations for link text clusters, redundant links, and similar average and minimum font sizes (10pt and 9pt).

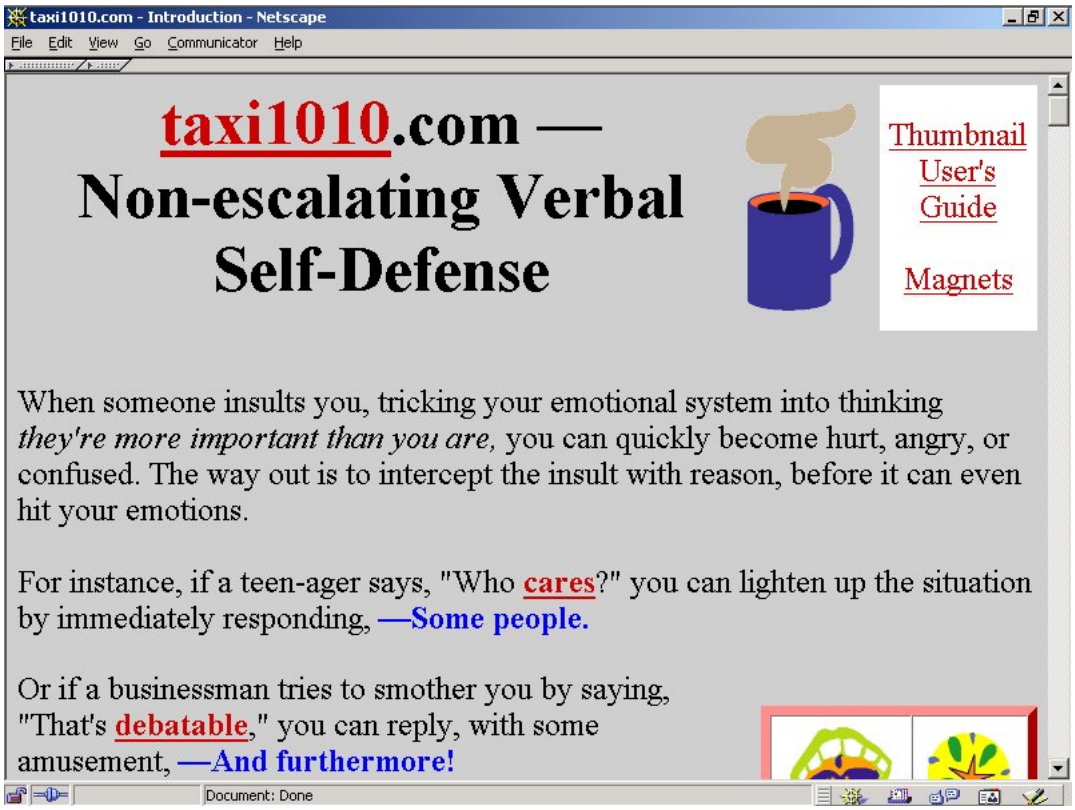


Figure 6.4: Poor page exhibiting several key properties of the class: italicized body text, repeated image links (images at the bottom right also appear in an area that is not visible), and minimal link text clustering.

of the decision tree measures were significantly different among clusters, except for the maximum graphic height.

Nine of the top ten measures are associated with the amount of text on a page, including the word count, good word count, HTML bytes, and vertical scrolls. Hence, the second and third clusters could be characterized as representing large and small pages. The large-page cluster is consistent with a group identified in a prior study by the author [Ivory *et al.* 2001], while the small-page cluster is consistent with two groups identified in the same study – low and medium word count pages. The remaining top ten measure – table count – distinguishes pages in the first cluster as ones that are highly formatted. Pages in the formatted-page cluster contain on average 120 more words than pages in the small-page cluster.

Pages closest to the centroid of each cluster are depicted in Figures 6.5, 6.6, and 6.7. All of the pages exhibit most of the properties previously discussed for good pages. Pages in the small-page and large-page clusters are similar in many ways, except for the amount of text on a page. ANOVAs contrasting these two clusters revealed that they are similar on 13 measures – the number of meta tag and good meta tag words, script bytes, minimum color use, good page title words, use of standard link colors, horizontal scrolls, graphical links, download time, and Bobby approval and priority 2 errors.

Pages in the formatted-page cluster are quite distinct from pages in the small-page and large-page clusters. They use more text positioning and columns, tables, and text color and panel color combinations. They also contain more graphics and redundant graphics, graphical ads, and have smaller minimum image widths and heights; correlations suggest that many of these graphics are possibly for organizing pages. Pages in this cluster also contain more interactive objects and colors. The example good page presented in Figure 6.2 belongs to this cluster.

6.8 Page Type Quality

The goal of this section is to show differences when the page type – home, link, content, form, or other – is included in the analysis. This analysis does not consider content categories. Table 6.15 summarizes the analyzed data.

Linear discriminant analysis was used to distinguish good, average, and poor pages within each page type, yielding an overall accuracy of 82%. Table 6.16 summarizes the accuracy of each page type model. These models are 7–15% less accurate than the overall page quality model. Recall that page types are predicted by a decision tree model with 84% overall accuracy (see Section 5.11). It is possible that mispredicted pages within each page type category may have negatively impacted model development.

ANOVAs revealed that the top ten predictor variables varied across page types, although several predictors from the overall page model were often among the top ten, including the interactive object count, minimum font size, italicized body word count, minimum color use, graphic ad count, and Bobby and Weblint errors. Key predictors in each page type model are discussed below. An effort was made to develop good page clusters for each of the page types, but the number of good pages for each type was inadequate.

Home Pages. The top ten measures for classifying good, average, and poor home pages include the following: graphic ad count (graphic element); object count, Weblint errors, and Bobby priority 2 errors (page performance); bad panel color combinations (page formatting); link and internal link counts (link element); italicized and exclaimed body word counts (text formatting); and link word count (text element). Good home pages contain considerably more links (internal links in particular) and a corresponding higher number of link words

Measure	Mean			Std. Dev.			F val.	Rank
	FP	LP	SP	FP	LP	SP		
Text Element Measures								
Word Count	360.6	849.2	215.7	194.7	221.2	140.2	1726.5	1
Good Word Count	203.1	449.3	115.9	110.5	112.8	74.5	1690.6	2
Body Word Count	200.3	621.1	164.4	138.8	196.1	132.4	1317.7	5
Good Body Word Count	92.4	303.5	79.2	70.6	94.5	65.9	1314.0	6
Fog Big Word Count	23.7	79.1	21.7	20.5	36.4	23.0	730.7	11
Good Link Word Count	80.0	62.3	27.8	41.9	47.2	26.5	364.9	22
Display Word Count	23.3	29.4	9.8	17.4	20.3	12.1	251.8	32
Exclamation Point Count	1.8	1.5	0.8	1.6	1.7	1.2	96.0	52
Good Meta Tag Word Count	23.7	12.0	12.7	25.3	20.6	22.7	38.6	60
Meta Tag Word Count	32.0	16.7	17.3	34.5	28.4	31.2	36.0	61
Good Page Title Word Count	2.8	3.3	3.2	1.6	1.7	1.7	16.8	66
Link Element Measures								
Link Count	67.6	49.5	26.3	25.6	31.2	17.4	525.6	18
Text Link Count	46.7	35.2	16.0	22.3	25.4	13.7	427.6	21
Redundant Link Count	12.6	8.4	5.2	8.6	8.3	5.7	171.2	39
Link Graphic Count	17.3	10.3	9.8	8.9	8.4	8.2	135.3	46
Graphic Element Measures								
Graphic Count	45.9	20.9	17.5	20.3	17.9	14.5	477.2	19
Redundant Graphic Count	21.7	7.5	4.7	13.7	10.7	7.9	452.6	20
Graphic Ad Count	2.2	0.9	0.8	1.5	1.3	1.2	230.1	33
Graphic Link Count	14.7	7.5	7.5	8.2	7.3	7.1	164.7	42
Text Formatting Measures								
Text Column Count	8.2	3.8	2.5	3.7	3.3	2.3	598.5	14
Link Text	2.4	1.2	0.7	1.4	1.4	1.0	311.3	26
Cluster Count								
Sans Serif Word Count	284.4	422.7	131.5	197.0	343.4	140.1	283.0	27
Text Cluster Count	3.7	3.0	1.2	2.4	2.6	1.5	278.1	29
Display Color Count	2.2	1.7	1.1	0.9	0.8	0.8	261.9	30
Text Positioning Count	5.8	3.7	2.3	3.5	3.4	2.5	214.1	34
Emphasized Body Word Count	56.1	102.0	33.6	51.3	73.1	48.2	202.8	36
Capitalized Body Word Count	1.5	3.5	1.2	2.3	2.9	2.1	136.6	44

Table 6.12: Means and standard deviations for the 3 clusters of good pages – formatted-page (FP), large-page (LP), and small-page (SP) (Table 1 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution (Rank column); the rank reflects the size of the F value.

Measure	Mean			Std. Dev.			F val.	Rank
	FP	LP	SP	FP	LP	SP		
Text Formatting Measures								
Bolded Body Word Count	12.6	20.9	7.3	14.9	19.6	13.5	108.9	50
Italicized Body Word Count	0.5	1.0	0.3	0.9	1.0	0.7	98.6	51
Serif Word Count	66.5	167.8	76.7	113.6	178.0	111.0	72.9	56
Colored Body Word Count	26.1	21.6	12.3	24.8	28.2	21.2	55.4	58
Average Font Size	10.2	11.0	10.8	1.0	1.2	1.2	52.1	59
Exclaimed Body Word Count	5.5	5.4	3.1	7.0	7.3	5.6	29.9	64
Minimum Font Size	8.9	9.0	9.0	0.3	0.2	0.2	7.3	70
Link Formatting Measures								
Standard Link Color Count	0.9	1.2	1.2	1.0	1.2	1.2	9.9	68
Graphic Formatting Measures								
Minimum Graphic Width	3.1	24.9	30.9	7.0	33.5	35.2	136.0	45
Minimum Graphic Height	2.3	10.8	13.3	4.1	14.1	15.0	116.8	48
Maximum Graphic Width	493.2	429.5	414.4	130.9	191.2	206.4	30.6	63
Page Formatting Measures								
Vertical Scrolls	2.1	3.8	1.2	1.2	1.3	0.9	748.8	9
Page Pixels	888K	1.4M	557K	393K	449K	284K	736.2	10
Color Count	11.1	7.8	6.8	2.6	2.2	1.7	687.7	12
Good Text Color Combinations	5.7	3.4	2.4	2.1	2.1	1.5	533.9	17
Bad Panel Color Combinations	2.1	0.9	0.5	1.3	1.2	1.0	312.3	25
Font Count	6.9	6.8	4.7	2.0	2.2	1.7	281.5	28
Good Panel Color Combinations	1.0	0.4	0.2	0.8	0.7	0.6	197.7	37
Interactive Object Count	5.7	2.6	2.2	3.8	3.3	3.3	169.5	41
Browser-Safe Color Count	6.0	5.0	4.7	1.8	1.4	1.3	129.6	47
Minimum Color Use	2.5	3.9	3.8	2.3	3.2	3.3	32.4	62
Horizontal Scrolls	0.0	0.1	0.1	0.2	0.3	0.3	10.3	67

Table 6.13: Means and standard deviations for the 3 clusters of good pages – formatted-page (FP), large-page (LP), and small-page (SP) (Table 2 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution (Rank column); the rank reflects the size of the F value.

Measure	Mean			Std. Dev.			F val.	Rank
	FP	LP	SP	FP	LP	SP		
Page Performance Measures								
Visible Unique Link Text Terms	99.2	270.6	75.6	75.6	66.6	47.2	1529.7	3
All Unique Link Text Terms	113.5	287.9	86.7	82.9	75.3	51.9	1335.3	4
HTML Bytes	25.1K	19.9K	9.4K	7.9K	8.3K	5.1K	986.9	7
Table Count	15.3	8.0	4.9	5.5	5.7	3.5	821.9	8
Visible Page Text Terms	189.3	520.6	195.0	152.1	158.2	156.4	646.6	13
All Page Text Terms	234.6	569.9	241.1	177.1	176.1	162.8	570.0	15
Weblint Errors	73.0	34.1	17.5	37.1	34.5	20.8	569.1	16
Bobby Browser Errors	20.6	13.0	11.2	6.6	7.0	5.9	357.1	23
Visible Page Text Score	126.6	239.8	111.8	106.7	105.2	83.0	259.8	31
Visible Page Text Hits	33.3	49.8	23.2	28.5	24.8	16.7	207.0	35
Bobby Priority 1 Errors	2.0	1.3	1.1	1.0	1.0	0.8	173.6	38
Object Count	3.3	1.6	1.3	2.2	2.2	1.7	170.5	40
All Page Text Hits	43.3	62.8	33.4	34.7	29.7	22.8	160.6	43
All Page Text Score	141.1	126.3	138.2	106.3	101.5	114.2	9.3	47
Bobby Priority 2 Errors	4.7	3.8	3.7	1.2	1.3	1.1	114.1	49
Page Title Terms	2.8	4.8	4.0	2.3	2.4	2.3	85.8	53
Script File Count	1.1	0.5	0.2	1.8	1.2	0.7	84.6	54
Unique Page Title Terms	2.8	4.7	3.9	2.3	2.3	2.2	83.2	55
Script Bytes	1.8K	1.0K	940.2	1.6K	1.4K	1.2K	68.1	57
Bobby Approved	0.1	0.2	0.2	0.3	0.4	0.4	29.7	65
Download Time	14.5	16.2	16.4	9.6	8.2	8.2	8.1	69

Table 6.14: Means and standard deviations for the 3 clusters of good pages – formatted-page (FP), large-page (LP), and small-page (SP) (Table 3 of 3). All measures are significantly different (.05 level) and sorted within each category by their contribution (Rank column); the rank reflects the size of the F value.

Page Type	Good	Average	Poor	Total
Home	213	187	159	559
Link	680	581	479	1740
Content	721	752	721	2194
Form	234	217	146	597
Other	54	98	100	252
Total	1902	1835	1605	5342

Table 6.15: Number of pages used to develop the page type quality models. Four pages with missing Bobby measures were discarded by the discriminant classification algorithm.

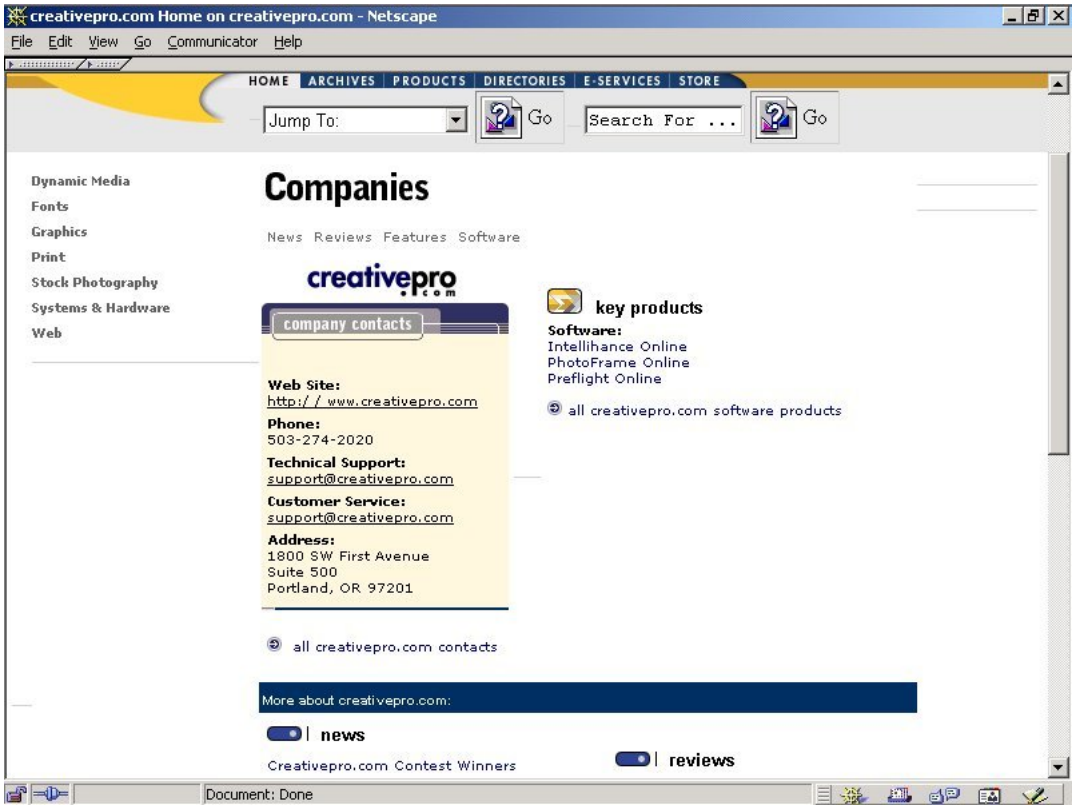


Figure 6.5: Page closest to the centroid of the formatted-page cluster (distance = 5.33 total standard deviation units of difference across all measures). Missing images are caused by a deficiency in the early version of the Site Crawler Tool.

Page Type	Sample Size	Classification Accuracy		
		Good	Average	Poor
Home	559	83.6%	80.2%	84.9%
Link	1740	80.1%	71.1%	78.3%
Content	2194	79.9%	74.2%	79.6%
Form	597	81.6%	77.0%	87.0%
Other	252	88.9%	76.5%	83.0%
Page Type Average		82.8%	75.8%	82.6%

Table 6.16: Classification accuracy for predicting good, average, and poor pages within page types.

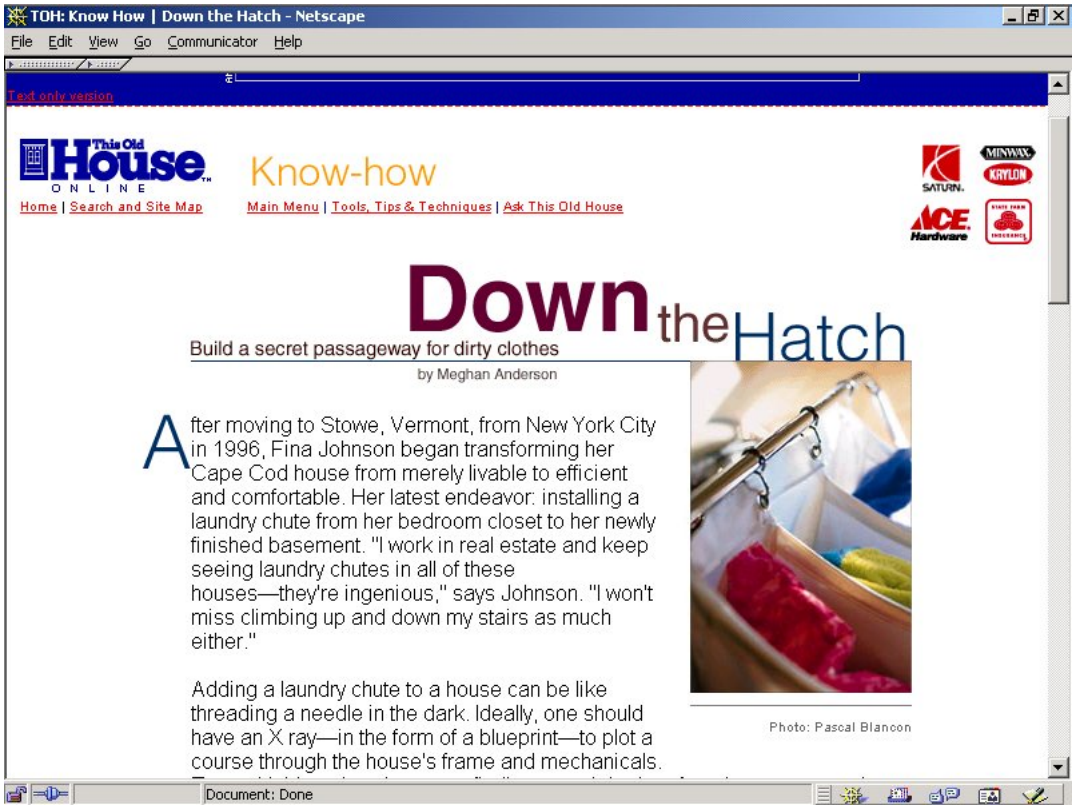


Figure 6.6: Page closest to the centroid of the large-page cluster (distance = 5.96 total standard deviation units of difference across all measures). Missing images are caused by a deficiency in the early version of the Site Crawler Tool.

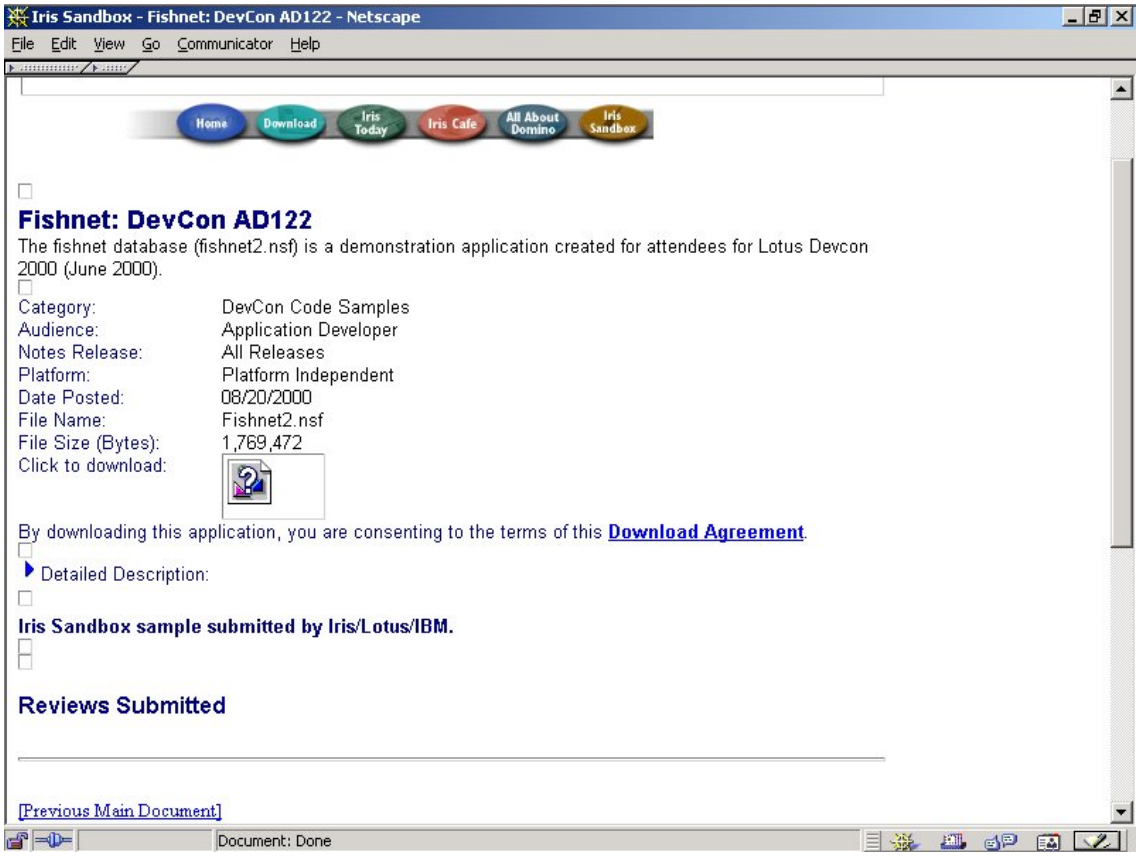


Figure 6.7: Page closest to the centroid of the small-page cluster (distance = 3.98 total standard deviation units of difference across all measures). Missing images are caused by a deficiency in the early version of the Site Crawler Tool.

than average and poor home pages. As was found for good pages overall, good home pages contain two graphical ads on average, while average and poor home pages contain an average of one graphical ad, respectively. Finally, good home pages use exclamation points to emphasize body text as opposed to italics; the converse is true for poor home pages.

Link Pages. The top ten measures for classifying link pages include: minimum font size, text cluster count, and text column count (text formatting); minimum color use and interactive object count (page formatting); Weblint errors (page performance); graphic and animated graphic ad counts (graphic element); and link and good link word counts (text element). Good link pages contain considerably more link and good link words than average and poor link pages. They also contain more text clusters, which suggests that text headings are used to organize groups of links. Good link pages also contain five text columns versus four and three for average and poor link pages, which suggests that links are organized into multiple columns. The last two claims were verified by examining a random sample of good link pages. Finally, good link pages are more likely to use an accent color unlike average and poor link pages.

Content Pages. The top predictor measures for classifying content pages include: minimum font size and italicized body word count (text formatting); minimum color use, good panel color combinations, and horizontal and vertical scroll counts (page formatting); spelling error count (text element); page title score and Bobby browser errors (page performance); and minimum graphic width (graphic formatting). Good content pages require an average of 2.4 vertical scrolls to read, while average and poor content pages require an average of 1.9 scrolls. This is because good content pages contain considerably more words than pages in the other categories; the difference is significant. Good content pages also require slightly more horizontal scrolls (.14 versus .1); however, horizontal scrolling is typically not required for content pages in any of the classes.

Average and poor content pages are more likely to contain good panel color combinations than good content pages suggesting that good content pages minimize colored areas on the page. Good content pages are also less likely to use page titles that are similar to source page titles suggesting the use of unique page titles among pages. Finally, good content pages appear to contain two spelling errors versus average and poor content pages that contain one spelling error. Inspection of a random sample of ten pages in each class revealed that most spelling errors (according to the Metrics Computation Tool) on good content pages are due to the use of jargon and abbreviations, such as cyberspace, messaging, groupware, busdev, and imusic. On the other hand, spelling errors on average and poor content pages tended to be true errors.

Form Pages. The top ten measures for classifying good, average, and poor form pages include the following: interactive object count (page formatting); minimum font size, body color count, and italicized body word count (text formatting); standard link color count (link formatting); graphic ad count (graphic element); Bobby browser and Weblint errors (page performance); minimum graphic width (graphic formatting); and graphic word count (text element). Many of the differences among form pages in the three classes mirror differences found with the overall page quality model. In addition, good form pages contain an average of eight interactive objects, while average and poor form pages contain an average of six interactive objects, respectively. Good form pages also use fewer than two body text colors unlike average and poor form pages that use more than two body text colors.

Content Category	Good	Average	Poor	Total
Community	542	365	216	1123
Education	518	433	391	1342
Finance	232	127	241	600
Health	172	457	299	928
Living	267	211	222	700
Services	171	242	236	649
Total	1902	1835	1605	5342

Table 6.17: Number of pages used to develop the content category quality models. Four pages with missing Bobby measures were discarded by the discriminant classification algorithm.

Content Category	Sample Size	Classification Accuracy		
		Good	Average	Poor
Community	1123	91.5%	87.9%	85.6%
Education	1342	87.8%	83.6%	85.7%
Finance	600	98.3%	98.4%	96.7%
Health	928	89.0%	93.4%	94.3%
Living	700	90.6%	89.1%	90.5%
Services	649	95.3%	95.0%	95.3%
Content Category Average		92.1%	91.2%	91.4%

Table 6.18: Classification accuracy for predicting good, average, and poor pages within content categories.

Other Pages. Recall that this page type broadly represents all remaining graphical (e.g., splash pages, image maps, and Flash) and non-graphical (e.g., blank, under construction, error, applets, text-based forms, and redirect) pages; thus, the pages have widely different features. The top ten measures for classifying other pages include the following: minimum color use and neutral text color combinations (page formatting); all link text hits, visible link text hits and score, all page text hits, visible page text hits (page performance); minimum graphic width (graphic formatting); and italicized and bolded body word counts (text formatting). Many of the top predictor measures are associated with the quality of scent between the source page's text and link text and the destination page's text. Good other pages have fewer common words with source link and page text than average pages but more common words than poor pages on all of the scent measures. Good other pages are more likely to contain neutral text color combinations than average and poor other pages. However, the three classes contain an equal number of good text color combinations, which suggests that good other pages use multiple text color combinations. Finally, good other pages contain about three bolded body words, while average and poor other pages contain six and twelve, respectively.

6.9 Content Category Quality (Pages)

The goal of this section is to show differences when the content category – community, education, finance, health, living, or services – is included in the analysis. The page type is not considered. Table 6.17 summarizes the analyzed data for each content category.

Linear discriminant analysis was used to distinguish good, average, and poor pages within each category. The classification models have an overall accuracy of 91%; Table 6.18 summarizes the accuracy of the each model. The average accuracy for the content category models is about

7–15% higher than the models developed for page types in Section 6.8. ANOVAs computed over pages accurately classified by each model revealed that the top ten predictor variables varied across content categories, although several predictors from the overall page model as well as the page type models were often among the top ten, including the minimum font size, italicized body word count, minimum color use, and Bobby and Weblint errors. Key predictors in each content category model are discussed below.

Community Pages. The top ten predictor measures for classifying good, average, and poor community pages include the following: minimum font size and undetermined font count (text formatting); spelling error count (text element); script file count, object count, and page title hits and score (page performance); minimum color use (page formatting); non-underlined text links (link formatting); and animated graphic ad count. Average and poor community pages were more likely to use fonts that are not recognized as serif or sans serif fonts (i.e., fonts that are not in the extensive lookup tables) than good pages. Good community pages are more likely to use scripts and one animated graphic on pages. Good community pages are more likely to contain text links without visible underlines than poor community pages but less so than average community pages. Finally, good community pages tend to use more distinct page titles between pages than the average and poor pages.

Education Pages. Key measures for classifying education pages include: Bobby priority 2 and browser errors and object bytes (page performance); minimum font size and italicized body word count (text formatting); minimum color use, fixed page width use, and interactive object count (page formatting); graphic and good graphic word counts (text element). Good education pages are less likely to use objects, such as applets, than poor education pages; there is no significant difference between good and average education pages. Good education pages are more likely to use a fixed page width (typically controlled by tables) than average and poor education pages. Good education pages contain about three interactive objects, while average and poor pages contain two and one, respectively. Similarly to other good pages, good education pages were slightly more likely to contain one graphical ad; however, this measure did not play a major role in classifying pages.

Finance Pages. The top ten predictor measures for classifying finance pages include the following: HTML file count, script bytes, and page title hits and score (page performance); serif font count, search object count, good text color combinations, and internal stylesheet use (page formatting); good graphic word count (text element); and non-underlined text links (link formatting). There are considerable differences in formatting for good finance pages compared to average and poor finance pages. Good finance pages consist of multiple HTML files due to the use of external stylesheets; they are also more likely to use internal stylesheets. Good finance pages contain an average of 1.1K bytes (σ = 1.2K) for scripts, while average and poor finance pages contain 1.9K (σ = 1.8K) and 883 bytes (σ = 812), respectively. Good finance pages use more distinct page titles between pages. They use more good text color combinations and search objects than average pages but fewer than poor pages; there was no difference in the number of bad and neutral text color combinations. Finally, good finance pages are more likely to contain text links without visible underlines and less likely to use serif fonts.

Health Pages. The top ten predictor measures for classifying good, average, and poor health pages include the following: Bobby priority 1 and browser errors, Weblint errors, and object count (page performance); italicized body word count (text formatting); link, link graphic,

internal, and redundant link counts (link element); and text column count (page formatting). There are several key differences in links on the good, average, and poor health pages. Specifically, good health pages contain an average of 48 links, while average and poor pages contain 33 and 28, respectively. Furthermore, good health pages contain more image, internal, and redundant links than average and poor health pages. Good health page also start text in about five different places on the page, while average and poor pages start text in three and two different places, respectively. This suggests that multiple columns are used to layout links on the page, which was confirmed by a higher link text cluster count for good health pages. Good health pages are also more likely to use scripts.

Living Pages. The top ten measures for classifying living pages include: minimum font size (text formatting); self containment and search object count (page formatting); Weblint errors (page performance); link and standard link color counts (link formatting); minimum graphic width and height (graphic formatting); animated graphic ad count (graphic element); and internal link count (link element). Good and average living pages use an average of three colors for links, while poor living pages use four. However, good living pages are less likely to use standard (browser default) link colors than average and poor living pages. They typically contain one animated graphical ad, one search form, and considerably more internal links than pages in the other classes. Good living pages are slightly less self-contained (i.e., the page can be rendered solely with the HTML code and associated images as opposed to requiring stylesheets, scripts, etc.) than average pages, but more so than poor pages; the self-containment measure on good living pages is largely due to the use of scripts.

Services Pages. The top ten predictor measures for classifying services pages include the following: minimum color use, text column count, and search object count (page formatting); Weblint errors, Bobby approved, table count, and graphic file count (page performance); spelling error count (text element); link text cluster count (text formatting); and non-underlined text links (link formatting). Good services pages are more likely to be Bobby approved than average and poor pages; this is the only case where this result was found. These pages start text in more places and use more link text clusters than average and poor pages. Good services pages are also more likely to contain links without visible underlines and use fewer graphic files than average and poor services pages.

6.10 Overall Site Quality

The goal of this section is to present an overall view of highly-rated sites that does not consider content categories. Most site-level measures require data from at least five pages for computation. Thus, the analyzed data only consists of 333 sites – 121 good sites, 118 average sites, and 94 poor sites.

To assign sites into the good, average, and poor classes, the C&RT trained on 70% of the data was used. The resulting tree contains 50 rules and has an overall accuracy of 81% (see Table 6.2 for more details). The accuracy of site predictions is lower than that of the other page-level models most likely because of a smaller training set; it is also possible that the site-level measures or prediction method need to be improved. Figure 6.8 depicts example decision tree rules for classifying sites.

ANOVAs for correctly classified sites revealed that the sites only differed significantly on the maximum depth measure. Table 6.19 shows that the median and maximum breadths crawled on the good sites are slightly higher than for average and poor sites, although not significantly

if ((Page Performance Variation is missing OR (Page Performance Variation $\leq$ 90.2)) AND (Overall Variation is not missing AND (Overall Variation $\leq$ 14.49)) AND (Link Element Variation is not missing AND (Link Element Variation $\leq$ 29.195)) AND (Link Element Variation is missing OR (Link Element Variation $>$ 20.98)))

Class = Good

This rule classifies sites as good sites if they have: 90.2% or less variation in page performance; 14.49% or less variation across all measures; and a link element variation between 20.98% and 29.2%.

---

if ((Page Performance Variation is missing OR (Page Performance Variation $\leq$ 90.2)) AND (Overall Variation is missing OR (Overall Variation $>$ 14.49)) AND (Graphic Element Variation is missing OR (Graphic Element Variation $\leq$ 185.5)) AND (Text Element Variation is missing OR (Text Element Variation $>$ 51.845)) AND (Graphic Formatting Variation is not missing AND (Graphic Formatting Variation $>$ 81.1)) AND (Median Page Breadth is missing OR (Median Page Breadth $\leq$ 10)))

Class = Average

This rule classifies sites as average sites if they have: 90.2% or less variation in page performance; variation across all measures greater than 14.49%; graphical element variation of 185.5% or less; graphic formatting variation greater than 81.1%; text element variation greater than 51.85%; and a medium breadth of ten pages or fewer at each level.

---

if ((Page Performance Variation is not missing AND (Page Performance Variation $>$ 90.2)))

Class = Poor

This rule classifies sites as poor sites if they have variations in page performance greater than 90.2%.

Figure 6.8: Example decision tree rules for predicting site classes (Good, Average, and Poor).

Measure	Mean			Std. Dev.		
	Good	Average	Poor	Good	Average	Poor
Maximum Depth	1.75	1.81	1.94	0.43	0.40	0.24
Median Breadth	7.34	7.21	7.05	4.85	3.99	4.11
Maximum Breadth	9.14	8.95	8.80	4.85	3.99	4.11

Table 6.19: Means and standard deviations for site architecture measures. These measures possibly provide some insight about the information architecture on good, average, and poor sites.

different. This suggests that the information architectures of good and average sites emphasize breadth over depth [Larson and Czerwinski 1998; Zaphiris and Mtei 1997].

The lack of significant differences on all but one measure suggests that relationships among measures is very important for classifying sites into the three classes, more so than with page classification. Examining large, unique correlations between measures on accurately-classified sites revealed the following.

- Correlations between text element and text formatting variation on good sites suggest that text formatting is altered as the amount of text increases on pages. Good sites also have slightly more variation on both of these measures than average and poor sites. Text formatting variation is also correlated with the maximum and median breadth at each level and the number of pages crawled on the site, which provides further support that text formatting varies among pages in good sites.

- Average sites only had one unique correlation – between graphic element and overall element variation. The overall element variation considers the amount of variation across pages on text, link, and graphic element measures, including the number of good display text words, text links, and animated images. The graphic element variation measure considers a subset of measures examined for the overall element variation measure, namely the graphic element measures. The large correlation between these two measures suggests that the overall element variation predominantly reflects variation in graphic elements as opposed to the variation in text and link elements.

- There were thirteen unique correlations between measures on poor sites. Most of the correlations suggest that variations in formatting (text, link, graphic, and page formatting variation) play a major role in the overall variation and page performance variation measures as opposed to variation in elements (text, link, and graphic element variation). Poor pages tend to have less formatting variation than average and good sites, but they have slightly more variation in page performance and element variation.

One of the limitations of the overall site quality model is that it does not consider the quality of pages in its predictions, because it is based on a completely different set of measures. Consequently, it is possible for the overall site quality model to predict that a site is consistent with good sites, but the overall page quality model predicts that all of the pages in the site are consistent with poor pages. Recall that the assumption throughout this chapter is that Webby judges' ratings apply to the site as a whole as well as to all of the pages within the site. Hence, it is not possible to incorporate page quality into the site quality model at this time. To remedy this situation, the median computed over predictions for individual pages in the site is reported

Content Category	Good	Average	Poor	Total
Community	34	22	15	71
Education	29	30	24	83
Finance	15	6	14	35
Health	12	27	17	56
Living	20	17	10	47
Services	11	16	14	41
Total	121	118	94	333

Table 6.20: Number of sites used to develop the content category quality models.

Content Category	Sample Size	Classification Accuracy		
		Good	Average	Poor
Community	71	88.2%	59.1%	66.7%
Education	83	79.3%	80.0%	66.7%
Finance	35	73.3%	83.3%	71.4%
Health	56	50.0%	81.5%	82.4%
Living	47	90.0%	76.5%	60.0%
Services	41	81.8%	93.8%	35.7%
Content Category Average		77.1%	79.0%	63.8%

Table 6.21: Classification accuracy for predicting good, average, and poor sites within content categories.

by the Analysis Tool; the predictions are generated by the overall page quality model discussed in Section 6.6. The median page-level prediction can be considered in conjunction with the site quality prediction in assessing the quality of a site.

6.11 Content Category Quality (Sites)

The goal of this section is to present an overall view of highly-rated sites that considers content categories; Table 6.20 summarizes the analyzed data. The C&RT was used to develop models for classifying the 333 sites into the good, average, and poor classes within the six content categories. Table 6.21 summarizes the classification accuracy of the decision tree models; accuracy for predicting poor sites is lower in most cases due to fewer sites. ANOVAs for correctly classified sites did not reveal significant differences in measures. Future work will entail developing a larger sample size, especially of poor sites, in order to improve predictions. The analysis suggests that a minimum of 35 sites per class and content category is needed to improve accuracy.

Similarly to the overall site quality model, the Analysis Tool reports the median computed over predictions for individual pages in the site as a way to incorporate page quality into assessing site quality within each content category. The predictions are generated by the content category models for pages; these models are discussed in Section 6.9.

6.12 Summary

This chapter presented several models for predicting expert ratings of pages and sites and hopefully for assessing Web interface quality as well. Page-level models were developed for the six content categories and five page types in addition to a model that classifies pages across

content categories and page types. Page-level models were also developed for content category and page type combinations, although these models were not discussed in detail; this model building effort demonstrated that it was possible to develop highly-accurate models, provided there were at least 60 pages for a content category and page type combination. Similarly, site-level models were developed across content categories as well as within content categories. Due to a smaller sample of site-level measures, the site-level models were not as accurate as page-level models.

Several key correlations were highlighted by the page-level models, including the use of an accent color on good pages, the use of fonts smaller than 9 pt for copyright and footer text on good pages, and the use of italicized body text on poor pages. The page type models showed that the measures found to be important for predictions were relevant to the functional style of pages. For example, it was found that good form pages use more interactive objects than average and poor form pages. Similarly, it was found that good link pages use more links than average and poor link pages. Overall, the key predictor measures varied across the models suggesting that an exhaustive set of page-level measures, such as the ones developed, is necessary for accurate predictions. The analysis also suggests that a broader set of site-level measures needs to be developed to improve predictions. The maximum crawling depth was the only key predictor for good sites; this measure in conjunction with the breadth measures suggest that good sites emphasize breadth over depth, which has been suggested in the literature.

Although some characteristics of pages and sites were presented for each model, more work needs to be done to better understand the design decisions encapsulated in the developed profiles. This is especially important for future work on supporting automated critique of Web interfaces. Furthermore, the efficacy of the developed profiles needs to be established via user feedback; this will be addressed in the remaining chapters. Specifically, Chapter 7 suggests that there is a relationship between expert ratings and usability ratings, Chapter 8 demonstrates that the profiles can be used to assess and improve the quality of Web sites, Chapter 9 shows that users prefer pages and sites modified based on the profiles over the original ones, and Chapter 10 demonstrates that the profiles can be used to examine established Web design guidelines.

Chapter 7

Linking Web Interface Profiles to Usability

7.1 Introduction

Chapter 6 demonstrated that profiles of highly-rated Web interfaces could be developed based on key quantitative measures. However, it is not clear what these profiles represent – highly usable, aesthetically-pleasing or perhaps merely popular pages. Two studies were conducted by the author to provide some insight about what the profiles represent. The first study (discussed in this chapter) evaluates what went into developing the profiles of Web interfaces, namely the expert ratings. The second study (discussed in Chapter 9) evaluates the results of applying the Web interface profiles.

This chapter discusses a usability study conducted to determine the relationship between Webby judges' scores and ratings assigned by participants (non experts) who used sites to complete tasks. The goal of this study was to determine if judges' scores were consistent with usability ratings for sites at the extremes of the rating scale (i.e., sites with overall scores in the top and bottom 33% of Webby sites).

The study produced usability ratings for 57 Web sites that were used in the development of profiles in Chapter 6. Thirty participants completed the study and rated sites in two conditions: after simply exploring the site (referred to as perceived usability); and after exploring and completing three information-seeking tasks on the site (referred to as actual usability). Participants rated sites in both conditions using the WAMMI usability scale [Kirakowski and Claridge 1998]. The analysis focused on answering the following questions.

- Are Webby judges' scores consistent with perceived usability ratings?
- Are Webby judges' scores consistent with actual usability ratings?

Analysis of study data suggests some consistency between the Webby judges' scores and both the actual and perceived usability ratings. One of the major limitations of this study is that it was conducted at least six months after sites were reviewed by Webby judges. Hence, sites may have undergone major changes in the interim. It may have been possible to use the Internet Archive[1] to determine whether sites had changed; however, the site evaluation dates were unknown and this information was needed for the assessment. Despite measured consistency between judges' and participants' ratings, a strong conclusion about judges' scores reflecting usability cannot be

[1] Available at http://www.archive.org/.

made from this study. The methodology described in this chapter could be repeated in a more ideal setting to enable stronger conclusions to be made.

7.2 User Study Design

A Web site usability study[2] was conducted between November 11, 2000 and November 17, 2000, in accordance with guidelines established by the Committee for the Protection of Human Subjects (project 2000-1-36). Thirty participants completed a between-subjects experiment wherein they explored and rated 22 sites in two conditions: simply exploring the site; and completing information-seeking tasks after exploring the site. Participants rated each site in only one condition. However, they rated eleven sites after exploring them and the other eleven sites after exploring and completing tasks on them. A total of 57 sites were evaluated by participants using the WAMMI (Website Analysis and MeasureMent Inventory) [Kirakowski and Claridge 1998] usability scale.

7.2.1 Study Sites and Tasks

For the analysis, 60 sites were selected from the Webby Awards 2000 dataset studied in Chapter 6. Half of the sites were from the good sample (top 33% of reviewed sites), and the other half were from the poor sample (bottom 33% of reviewed sites). All of the sites fell within the top/bottom cutoffs discussed in Section 6.3.1; this was the case for the overall Webby score and the Webby factor (variable derived via principal components analysis to summarize the six rating criterion) across all content categories as well as within each content category. There was equal representation among the six categories (Community, Education, Health, Finance, Living, and Services). Furthermore, the selected sites met the following criteria: the site used the English language; the site was not implemented with Macromedia Flash; and the site did not require login. Three of the sites became unavailable or malfunctioned during the course of this study; thus, results are only reported for 57 sites. In some cases participants experienced technical difficulties; hence, responses were eliminated in these situations as well.

Three information-seeking tasks were developed for sites. First, sites within each of the six categories were explored to develop two general tasks, such as finding information about how to contact the company or the major product/service offered through the site. Table 7.1 contains the two general tasks developed for each content category. General tasks required participants to locate information that has been noted as essential to good Web design practices in the literature [Fogg *et al.* 2000; Nielsen 2000; Sano 1996]. Each site was then explored to identify a site-specific information-seeking task. These tasks were non obvious, required participants to follow at least five links through the site, and were comparable to tasks chosen for the other sites in the content category. Tables 7.2 and 7.3 summarize site-specific tasks.

A testing interface was developed using HTML forms, JavaScript, and PERL. Figure 7.1 depicts the screens for performing information-seeking tasks in the actual usability condition. In addition, a script was developed to generate 30 randomized experiment designs (i.e., the original 60 sites where randomly assigned to these experiments). Each experiment consisted of 22 sites, two of which were for training. The order of site exploration was randomized as well as the presentation of the three tasks. Two pilot studies were conducted to improve the final study design, testing interface, and testing materials.

[2] Rashmi Sinha and Marti Hearst provided valuable input into the study design, assisted with recruiting participants, and facilitated several testing sessions. Sinha also conducted a preliminary analysis of the study data.

Content Category	Information to locate in the site
Community	How to contact the company What topics are discussed on this site
Education	What educational products/services are offered through this site What topics/disciplines does this site address
Finance	What is the major financial product/service offered through this site Get a sense for whom this site is meant for
Health	What topics are discussed on this site Find contact information for the site
Living	Get a sense for whom this site is meant for What is the major product/service offered through this site
Services	What is the major product/service offered through this site How to contact the company

Table 7.1: General information-seeking tasks for each content category.

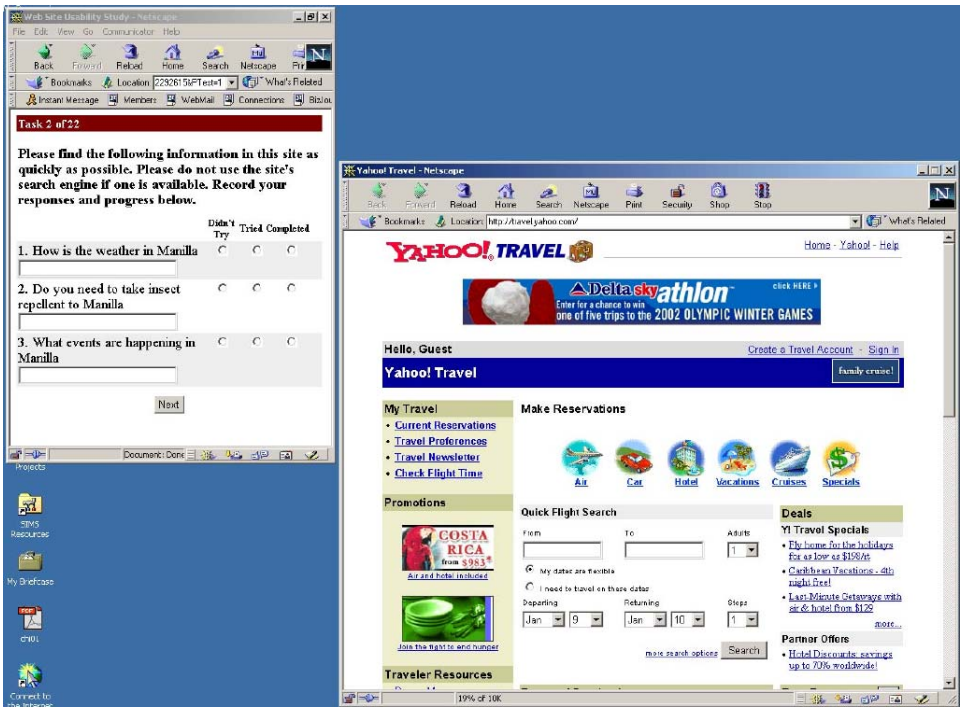


Figure 7.1: Testing interface for completing the information-seeking tasks in the actual usability condition. The smaller browser window provides instructions, and the larger window displays the site.

Id	Rating	Information to locate in the site
Community		
1_242	Good	Article on computer privacy
1_250	Good	A good age to start teaching children to swim
1_255	Good	The height of the second tallest mountain in California
1_261	Good	How to find donated items for your local charity
1_262	Good	Discussions on rising heat costs
1_003	Poor	Details about the movie The Cell
1_006	Poor	Details about the Age of Attention
1_007	Poor	Details about recent findings in Gamla
1_009	Poor	List of items you need for playing rugby
1_015	Poor	Details about horsepower
Education		
2_205	Good	Different kinds of computers
2_208	Good	Causes of drought
2_209	Good	School voucher issues
2_217	Good	Courses on snowboarding
2_218	Good	Details about Madagascar
2_012	Poor	Benefits of water birth
2_013	Poor	Leadership case studies
2_016	Poor	The student who designed the Iceman poster
2_017	Poor	Requirements for being a student anchor
2_018	Poor	Details about the Brazil training stress
Finance		
3_087	Good	Status of car sales in the US
3_098	Good	Basic investment options
3_101	Good	Details about student loans
3_102	Good	Investment strategies for college students
3_103	Good	Details about Roth IRAs
3_002	Poor	How the market affects this company's funds
3_014	Poor	How a loan officer evaluates your credit report
3_018	Poor	How Theresa Pan became a billionaire
3_019	Poor	Return for the best stock picked by Sam Isaly
3_022	Poor	How education influences compensation of financial executives

Table 7.2: Site-specific information-seeking tasks (Community, Education, and Finance).

Id	Rating	Information to locate in the site
Health		
4_127	Good	Techniques for coping with stress
4_128	Good	The vaccine for Lyme disease
4_129	Good	Flu shots for elders
4_130	Good	Impact of second-hand smoke on asthma sufferers
4_132	Good	Latest findings on job-related depression
4_005	Poor	Links to online health statistics
4_008	Poor	Details about e-pharmacy.MD
4_018	Poor	Issues that are best suited for online therapy
4_020	Poor	Treatment for common colds
4_021	Poor	Advice for seeking a second medical opinion
Living		
5_137	Good	Details about The Zone diet
5_138	Good	Using sesame seed in recipes
5_139	Good	Taking a virtual tour of New York
5_145	Good	Cost for copyrighting software
5_002	Poor	Menu for an inexpensive dinner party
5_005	Poor	Recipe for Chili Cornmeal Biscuits
5_008	Poor	Price list for California wines
5_014	Poor	Creating a wish list
5_021	Poor	Whether you can recycle the tray from your food package
Services		
6_115	Good	How to receive faxes online
6_116	Good	Cost for a recycled front door window for a 1992 Honda Civic DX Hatchback
6_122	Good	Creating a wish list
6_132	Good	Use of foods to remedy medical problems
6_008	Poor	The ad for a postcard designed for students
6_009	Poor	Details about Desktop Drive
6_018	Poor	Cost for developing a web site
6_019	Poor	The newsletter issue that discusses Internet attitudes and usage

Table 7.3: Site-specific information-seeking tasks (Health, Living, and Services).

Id	Age	Gen	Education	CmpExp	IExp	IUse	EngExp	EvalExp
1	18-25	F	College Grad.	Average	Average	>10	Expert	Average
2	18-25	M	Some College	Average	Average	3-5	Average	Average
3	18-25	F	Some College	Beginner	Beginner	1-2	Average	Beginner
4	18-25	M	Some College	Expert	Expert	>10	Expert	Expert
5	18-25	F	Some College	Expert	Expert	>10	Expert	Expert
6	18-25	M	Some College	Expert	Expert	>10	Expert	Beginner
7	18-25	M	Some College	Average	Average	>10	Average	Average
8	18-25	F	Some College	Expert	Expert	>10	Expert	Expert
9	18-25	M	Some College	Average	Average	>10	Average	Average
10	18-25	M	Some College	Average	Expert	>10	Expert	Average
11	18-25	F	Some College	Average	Average	1-2	Expert	Beginner
12	18-25	F	Some College	Average	Average	1-2	Expert	Average
13	18-25	F	≤High School	Average	Average	6-10	Expert	Average
14	18-25	M	Some College	Expert	Expert	>10	Expert	Expert
15	18-25	F	Some College	Average	Expert	>10	Expert	Average
16	26-35	M	Some College	Average	Average	1-2	Expert	Beginner
17	18-25	F	Some College	Average	Average	3-5	Expert	Average
18	18-25	F	Some College	Average	Average	>10	Expert	Average
19	18-25	F	Some College	Average	Average	6-10	Average	Beginner
20	18-25	F	≤High School	Average	Average	1-2	Average	Average
21	18-25	F	Some College	Average	Average	>10	Average	Beginner
22	26-35	M	Some College	Average	Expert	>10	Expert	Average
23	18-25	F	Some College	Average	Average	3-5	Expert	Beginner
24	18-25	F	Some College	Expert	Expert	>10	Expert	Expert
25	18-25	F	Some College	Average	Average	6-10	Average	Beginner
26	36-45	F	Some College	Average	Average	6-10	Expert	Average
27	18-25	F	Some College	Average	Average	6-10	Expert	Expert
28	18-25	F	Some College	Average	Expert	>10	Average	Average
29	18-25	M	≤High School	Expert	Expert	6-10	Expert	Expert
30	18-25	F	Some College	Average	Average	3-5	Average	Beginner

Table 7.4: Summary of participants' demographic information. Participants provided their age and gender (Gen) and described their proficiency with computers (CmpExp), the Internet (IExp), the English language (EngExp), and evaluating Web site quality (EvalExp). Participants also reported the number of hours they spend using the Internet weekly (IUse).

7.2.2 Participants

Study participants were recruited through undergraduate classes and sign-up sheets posted in campus buildings. Thirty participants, primarily UC Berkeley undergraduates, completed the study and were each compensated with $30. Participants answered demographic questions prior to starting the study. Specifically, participants provided their age, gender, as well as information about their education background, computer and Internet experience, the number of hours they use the Internet weekly, English experience, and Web site evaluation experience. Table 7.4 summarizes this information.

All but three of the participants were in the 18–25 age group, and all but four of the participants were undergraduates. Two thirds of the participants were females. Only one participant was a novice computer and Internet user; the remaining participants described themselves primarily as average users. Half of the participants reported using the Internet for over ten hours

a week. Half of the participants also reported having some experience evaluating the ease of use of Web sites. Finally, all of the participants were experienced with the English language.

Based on the demographic information, all of the participants appear to have been appropriate for this study.

7.2.3 Testing Session

The study consisted of a 150-minute session wherein participants interacted with 22 sites; the first two sites were for training. Participants were initially given an instruction sheet (see Figure 7.2) that provided an overview of the study procedure. After reviewing the instruction sheet, participants completed a statement of informed consent and moved to a computer station for completing the study. The study interface requested demographic information and assigned each participant to one of the 30 experiment designs. The interface subsequently guided participants through two types of tasks as discussed below.

- **Rate the site without completing information-seeking tasks.** Participants were instructed to initially explore the site for several minutes. Then, participants were asked to rate the site along a number of criteria and provide freeform comments about the site. This task type assessed the *perceived* usability of a site.

- **Rate the site after completing information-seeking tasks.** Participants were instructed to initially explore the site for several minutes. Then, participants were presented with three tasks and asked to locate the information as quickly as possible without using the site's search engine. The testing interface presented tasks in a randomized order and provided text fields for participants to enter the information that they found; there were also radio buttons for participants to record their progress on each task (didn't try, tried, and completed). The testing interface also recorded the total time spent on the three tasks. Finally, participants were asked to rate the site along a number of criteria and provide freeform comments about the site. This task type assessed the *actual* usability of the site, since participants attempted to use the site.

During the testing session, participants alternated between completing these two types of tasks on the 22 sites. The testing interface timed participants during the exploration and task completion phases. A message window appeared whenever participants spent more than five minutes on either phase.

For the rating phase, participants responded to 20 statements from the WAMMI [Kirakowski and Claridge 1998] questionnaire. WAMMI was the only validated usability scale for Web sites at the time of this study. Responses to WAMMI statements were aggregated into six scales: attractiveness, controllability, efficiency, helpfulness, learnability, and global usability (see Section 7.5). Jurek Kirakowski, Director of the Human Factors Research Group in Ireland, converted participant responses into WAMMI scales for this study. This conversion process also entails normalizing computed scales against a database of other sites that have been assessed with the questionnaire. The reported WAMMI scales are percentiles that reflect how the site's rating contrast to other sites that have been rated with the WAMMI questionnaire.

Participants were also asked about their confidence in responses given to WAMMI statements as well as their opinion about the appropriateness of tasks. Participants responded to all statements using a 5-point Likert scale, ranging from strongly agree (1) to strongly disagree (5); this order was required for consistency with the WAMMI questionnaire. Figure 7.3 depicts all of the statements used during the rating phase. The testing interface presented WAMMI statements

Web Site Usability Study Instructions

I. Overview of the Study
The purpose of this study is to get user feedback on the ease of use of a collection of web sites. Basically, you will explore the site and provide responses to a number of statements about the site's ease of use. There is no risk to you, and you will be compensated at a rate of $12/hour for your participation.

Please read and sign the informed consent form and return to the tester.

II. Overview of the Study Procedure
General goal:
Rate web sites. There will be two kinds of tasks. (a) Find some information on the web site. After you have completed all of the information-seeking tasks, then rate it. (b) Rate the web site after only exploring it.

Tasks
(a) **Rate web site after completing information-seeking tasks on it.**
In the first part of the task, you will be asked to just explore the site. Look around and get familiar with the content, layout, navigation, etc. of the site. **Please do not spend more than a few minutes on the site.**
In the second part of the task, you will be required to complete some information-seeking tasks on the site. Getting familiar with the site in the first part of the task will help you prepare for this part. **You are asked to locate the information as quickly as possible.** We will time you on the task; also try to respond as accurately as you can.
General instructions for the task

- Follow links, don't use search.
- Stay on the site. If by chance you choose a link that takes you out of the site, come back and try to stay on the site.
- Do not spend more than a few minutes exploring the site; you will be reminded.
- The first task is for training.

(b) **Rate web site without completing tasks on it.**
You will go to a particular web site and explore it. Then rate the site. **Do not spend more than a few minutes exploring the site; you will be reminded.** The first task is for training.

Ratings:
Ratings are for 21 criteria on a scale. For each statement you will provide a response that reflects your agreement with the statement (Strongly Agree – Strongly Disagree). You can make any general (free form) comments that you might have. Finally, please note your prior experience with the site before this study.

Take breaks whenever you want to. If you need a break, try to take one in between two sites, rather than in the middle of completing tasks on a site. Remember not to spend too much time on one site; there are 22 sites in the study.

Figure 7.2: Instructions given to study participants.

in a randomized order to increase the likelihood that participants actually read statements before responding.

Both the computed WAMMI scales and the original responses to WAMMI statements were analyzed. Likert scales for positive statements were inverted such that positive responses resulted in higher scores (i.e., higher is better); this adjusts responses to statements 2, 3, 5, 6, 7, 8, 10, 11, 15, 17, 20, 21, and 22 as depicted in Figure 7.3. Overall, participants reported that they were highly confident with their responses (statement 21) and felt that the tasks were useful (statement 22). Figure 7.4 depicts distributions of responses to these two statements.

Participants had the opportunity to provide freeform comments and to record the number of times (never, 1–3 times, and 3 or more times) they had used a site prior to this study. In all but three cases, participants had never used the sites in the study. Two participants reported using a site more than three times, and another participant reported using a site 1–3 times. These responses were eliminated from the analysis, since they were potentially biased based on prior use. In addition to subjective ratings, the testing interface recorded timing information for the exploration, task completion, and rating phases.

7.2.4 Testing Environment

Participants completed the study in one of seven group sessions in UC Berkeley's School of Information Management and Science's second-floor computer lab. Participants worked individually at a computer station during these sessions. Computer stations had PCs running Microsoft Windows NT 4.0 with 128 MB of RAM. Stations also had 17" monitors (1280 x 1024 pixels) and high speed, campus Ethernet connections. Participants used the Netscape Navigator 4.7 browser. The testing interface resized the browser window to 800 x 600 pixels (equivalent to a 15" monitor), and sites were not cached in order to provide a realistic experience. User surveys have shown that over 50% of Internet users access sites with 800 x 600 monitor resolution and 56K and slower connection speeds [DreamInk 2000].

7.3 Data Collection

Several sites became unavailable during the course of this study, and in some cases participants were unable to rate sites due to time constraints; responses were eliminated for these situations. Responses were also eliminated for training sites and the three cases where participants had used sites prior to the study, since prior use may suggest a bias. Finally, one case was eliminated wherein the participant did not attempt any tasks in the actual usability condition.

The final dataset consisted of 550 cases where each case included participant's responses to the statements in Figure 7.3, computed WAMMI scales (attractiveness, controllability, efficiency, helpfulness, learnability, and global usability), Webby scores (content, structure and navigation, visual design, functionality, interactivity, and overall experience), and the Webby factor for the site, participant's demographic information, objective measures (e.g., exploration and rating time), and participant's comments. There were 271 cases for the actual usability condition (i.e., with information-seeking tasks) and 279 cases for the perceived usability condition (i.e., without information-seeking tasks). Table 7.5 summarizes the distribution of cases. All of the sites were rated in both the actual and perceived usability conditions. An average of five participants rated each site in the two conditions; thus, a site was rated by an average of ten participants overall.

Each of the 550 cases contained 66 fields of information as described below.

Rating Criteria

1. This web site has some annoying features. (-, annoy)
2. I feel in control when I'm using this web site. (+, control)
3. Using this web site for the first time is easy. (+, easy)
4. This web site is too slow. (-, slow)
5. This web site helps me find what I am looking for. (+, find)
6. Everything on this web site is easy to understand. (+, clear)
7. The pages on this web site are very attractive. (+, pretty)
8. This web site seems logical to me. (+, logical)
9. It is difficult to tell if this web site has what I want. (-, nofind)
10. This web site has much that is of interest to me. (+, interest)
11. I can quickly find what I want on this web site. (+, qfind)
12. It is difficult to move around this web site. (-, nonav)
13. Remembering where I am on this web site is difficult. (-, noremind)
14. Using this web site is a waste of time. (-, waste)
15. I can easily contact the people I want to on this web site. (+, contact)
16. I don't like using this web site. (-, nolike)
17. I get what I expect when I click on things on this web site. (+, expect)
18. This web site needs more introductory explanations. (-, nointros)
19. Learning to find my way around this web site is a problem. (-, nolearn)
20. I feel efficient when I'm using this web site. (+, effic)
21. I feel very confident about my responses to the previous statements regarding this web site. (+, conf)
22. These tasks enabled me to get an overview of the site. (+, taskuse, for information-seeking tasks only)

Figure 7.3: Rating criteria used during the study. The first 20 statements are from the WAMMI questionnaire. Statements are noted as positive (+) or negative (-) and a short descriptor is provided for each.

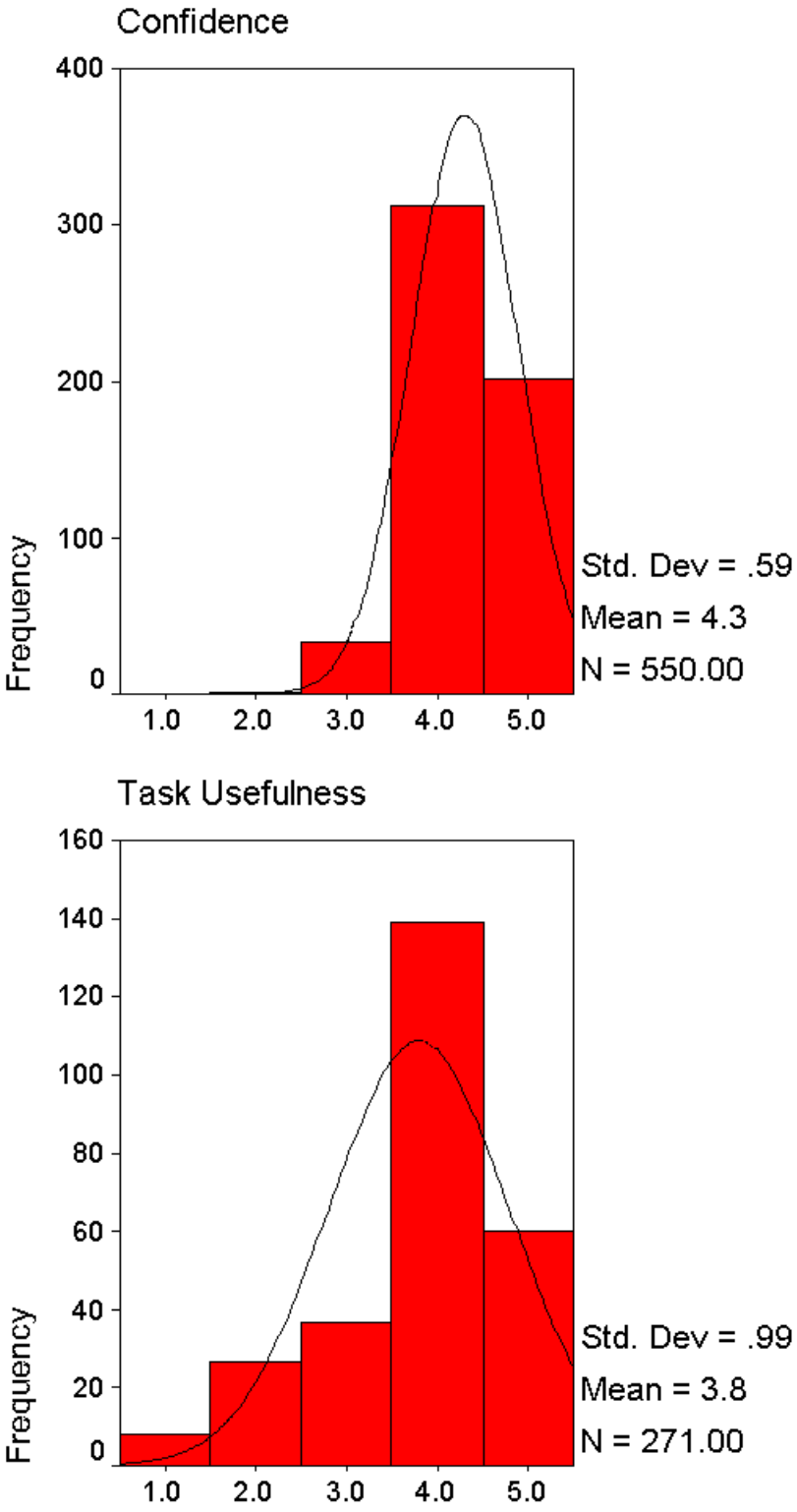


Figure 7.4: Study participants' confidence with responses to WAMMI statements (top graph) and reported usefulness of tasks in the actual usability condition (bottom graph). Responses range from low (1) to high (5).

Category	Good	Poor	Total
Actual Usability			
Community	25	25	50
Education	23	22	45
Finance	22	24	46
Health	24	25	49
Living	18	24	42
Services	20	19	39
Total	132	139	271
Perceived Usability			
Community	23	25	48
Education	25	25	50
Finance	24	25	49
Health	25	24	49
Living	19	25	44
Services	19	20	39
Total	135	144	279

Table 7.5: Number of cases (ratings) in the analyzed sample.

7.3.1 Basic Information

Site Identification: the site's category (Community, Education, Finance, Health, Living, or Services), URL, title, id, and whether the site belonged to the top or bottom 33% of reviewed sites (i.e., good or poor class).

Study Condition: participant id and experiment condition (actual or perceived usability).

Participant Demographics: participant's age, gender, educational background, computer experience, Internet experience, weekly Internet use, English experience, and Web site evaluation experience. Table 7.4 summarizes this information.

7.3.2 Objective Measures

Timing: site exploration time (exptime), time spent completing tasks in the actual usability condition (tasktime), and time spent rating the site in both conditions (ratetime). Total time spent on the site (exp+task) was also computed for analysis; this time is the same as site exploration time in the perceived usability condition.

Task Completion: whether the participant did not attempt to complete, attempted to complete, or completed each of the three tasks (t1suc, t2suc, and t3suc). The number of tasks that were not attempted (notry), not completed (nocomp), and completed (comp) were also tallied.

7.3.3 Subjective Measures

Responses to WAMMI Statements: subjective measures included the participant's responses to statements 1 through 20 of Figure 7.3 – annoy, control, easy, slow, find, clear, pretty, logical, nofind, interest, qfind, nonav, noremind, waste, contact, nolike, expect, nointros, nolearn, and effic – respectively. Positive statements were inverted as previously discussed.

Computed WAMMI Scales: attractiveness (wamattr), controllability (wamcon), efficiency (wameff), helpfulness (wamhelp), learnability (wamlearn), and global usability (wamglob).

Composite WAMMI Score: There was an attempt to compute a composite WAMMI score similarly to the computed Webby factor. However, there was little correlation among the six WAMMI scales. Although, there were large positive correlations between global usability and attractiveness, controllability, helpfulness, and learnability, there was a small negative correlation with efficiency. Hence, it was not possible to compute a single factor that could explain most of the variance in these scales. Instead, responses to the 20 statements were summed to represent a composite score (wamsum). Section 7.4 discusses this in more detail.

Webby Scores: content (content), structure and navigation (nav), visual design (vis), functionality (funct), interactivity (int), and overall experience (overall).

Composite Webby Score: the computed Webby factor (webbyfac); this measure was derived via principal components analysis over the six Webby scores.

Other Measures: confidence with responses to WAMMI statements (conf), usefulness of tasks in the actual usability condition (taskuse), and the number of times the participant had used the site prior to the study (prioruse).

Task Responses: answers and comments provided for each of the three information-seeking tasks (t1resp, t2resp, and t3resp) in the actual usability condition.

Comments: freeform comments on the site's ease of use.

7.3.4 Screening of Objective and Subjective Measures

The data was screened to replace extremely large and small outliers with the next smallest or largest values as appropriate for each of the objective and subjective measures; this is a standard statistical process used to remove potential errors in measurement from analysis data [Easton and McColl 1997; SPSS Inc. 1999]. Tests for normality and equal variances revealed that most of these measures did not follow a normal distribution, although most exhibited equal variances. Standard statistical analysis techniques assume these two conditions; hence, it was not possible to use parametric techniques with this dataset. Applying transformations to stabilize the data, such as square roots and reciprocals of square roots, were unsuccessful. Nonparametric analysis techniques were used during analysis, since they do not require data to satisfy the normality and equal variances conditions.

7.4 Developing a Composite WAMMI Score

When participants respond to multiple questions to provide subjective ratings, it is a common practice to summarize these responses with one factor. For example, the Questionnaire for User Interaction Satisfaction (QUIS) [Harper and Norman 1993] requires participants to rate an interface on 27 facets, the responses can then be summed to produce an overall measure of user satisfaction; it is also possible to aggregate subsets of responses into several interface factors, including system feedback and learning. For the second metrics study, principal components analysis was used to produce a composite Webby score – the Webby factor; this factor summarized judges'

Scale	wamattr	wamcon	wameff	wamhelp	wamlearn	wamglob
wamattr	1.00	**0.17**	-0.03	**0.16**	**0.11**	**0.49**
wamcon	**0.17**	1.00	**-0.35**	**0.45**	**0.17**	**0.55**
wameff	-0.03	**-0.35**	1.00	**-0.41**	**-0.18**	-0.07
wamhelp	**0.16**	**0.45**	**-0.41**	1.00	**0.11**	**0.64**
wamlearn	**0.11**	**0.17**	**-0.18**	**0.11**	1.00	**0.52**
wamglob	**0.49**	**0.55**	-0.07	**0.64**	**0.52**	1.00

Table 7.6: Correlations between the 6 WAMMI scales. Bold entries are significant.

ratings for the six criteria and made it possible to produce more accurate predictions than with the overall experience score [Ivory *et al.* 2001].

Similarly to the Webby factor, there was an attempt to compute a composite WAMMI factor based on the six scales. Table 7.6 shows correlations between pairs of WAMMI scales; Spearman correlation coefficients were computed, since this method is appropriate for nonparametric data. With the exception of the global usability scale (wamglob), there was only small to medium correlations between scales; correlations were surprisingly negative in some cases. The global usability scale had large positive correlations with all of the scales, except efficiency (wameff). Inconsistency among the scales is somewhat expected, since there were only five or fewer responses for each site in each of the two conditions. According to the developers of the WAMMI scales, scales typically require 20 or more responses to stabilize[3]. Given the instability of the scales, it was not possible to construct a single factor to summarize them. Several factor analysis approaches were used, such as principal components and maximum likelihood with and without rotation; this analysis could only produce two factors that explained 63% of the total variance in the scales.

A composite score was computed by summing responses to the 20 statements; this composite score, wamsum, ranges from 20 to 100 and naturally correlates with all of the statements. Figure 7.5 shows that most of the sums were towards the middle of the range.

7.5 Mapping Between WAMMI Scales and Webby Scores

As discussed in Section 7.3.1, subjective measures included the computed WAMMI scales (attractiveness, controllability, efficiency, helpfulness, learnability, and global usability) and the Webby scores (content, structure and navigation, visual design, functionality, interactivity, and overall experience). Kirakowski and Claridge [1998] claim that WAMMI scales were developed and validated through empirical studies, as is typically done with psychological scales. Although not verified, the Webby scores were developed based on consensus of the members of the International Academy of Digital Arts & Sciences. It is not clear how the academy members derived the scores or what instructions were given to judges to facilitate ratings.

Based solely on the descriptions of Webby scores and WAMMI scales, there appeared to be direct mappings between five criteria in the two rating schemes as depicted in Table 7.7. The learnability and interactivity criteria did not appear to be closely related. There are several key differences between the two ratings schemes:

- Webby scores are average ratings (typically for three judges), while WAMMI scales are computed from responses to individual statements and normalized against other assessed sites.

[3]Personal communication with Jurek Kirakowsi on December 2, 2000.

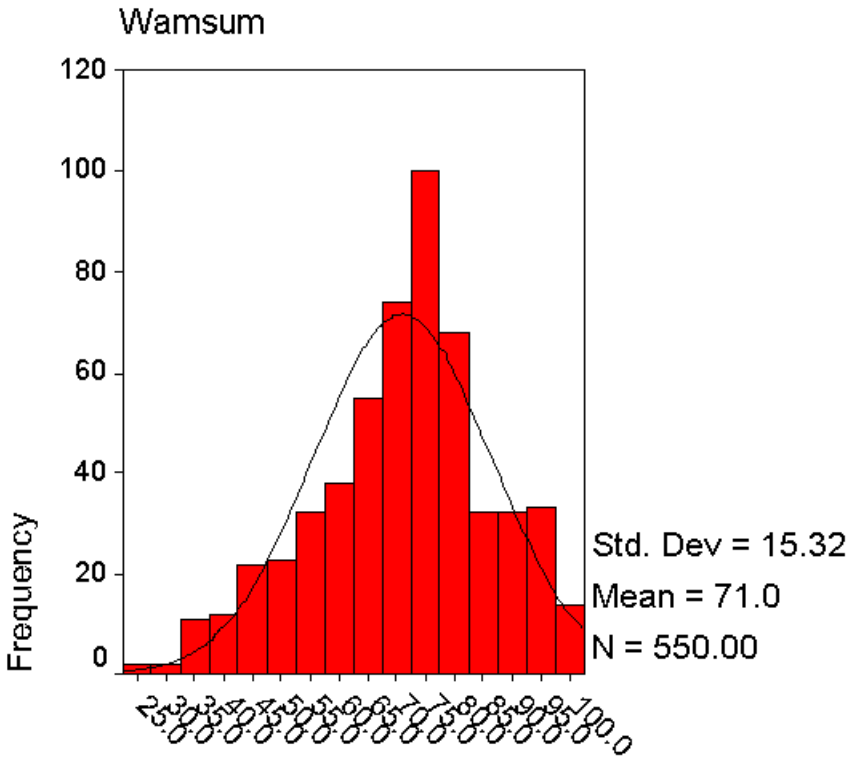


Figure 7.5: Wamsum scores (sums of responses to the 20 WAMMI statements) for the analyzed sample.

WAMMI Scales	**Webby Scores**
Helpfulness – "corresponds with the users' expectations about its content and structure..."	*Content* – "the information provided on the site..."
Controllability – "the users most probably feel they can navigate around it with ease..."	*Structure and Navigation* – "the organization of information on the site and the method in which you move through sections..."
Attractiveness – "An attractive site is visually pleasant..."	*Visual Design* – "appearance of the site..."
Efficiency – user "can quickly locate what is of interest to them and they feel that the web site responds..."	*Functionality* – "the site loads quickly, has live links, and any new technology used is functional and relevant..."
Global Usability – "a site must make it easy for users to access what they need or want..."	*Overall Experience* – "encompasses content, structure and navigation, visual design, functionality, and interactivity, but it also encompasses the intangibles that make one stay or leave..."
Learnability – "users feel they are able to start using the site with the minimum of introductions..."	–
–	*Interactivity* – "the way that a site allows a user to do something..."

Table 7.7: Description of the WAMMI usability scales and Webby scores in conjunction with mappings between these two rating schemes. Mappings are based solely on descriptions.

Criterion	Min.	Max.	Mean	Std. Dev.	Med.
Webby Scores					
content	1.3	10.0	6.4	2.5	7.0
nav	1.3	9.3	6.1	2.3	6.7
vis	1.3	9.0	5.5	2.1	6.0
funct	1.0	10.0	5.6	2.5	6.0
int	1.3	9.5	6.4	2.2	7.3
overall	1.0	10.0	5.9	2.5	5.3
WAMMI Scales					
wamattr	5.0	62.0	26.2	12.1	25.0
wamcon	2.0	65.0	15.8	14.5	11.0
wameff	4.0	61.0	29.6	13.1	27.0
wamhelp	2.0	83.0	27.8	20.9	19.0
wamlearn	4.0	71.0	28.8	15.4	24.5
wamglob	11.0	50.0	25.3	7.5	24.0

Table 7.8: Descriptive statistics for Webby scores and WAMMI scales.

Score	content	nav	vis	funct	int	overall
content	1.00	0.91	0.83	0.90	0.90	0.96
nav	0.91	1.00	0.90	0.90	0.93	0.93
vis	0.83	0.90	1.00	0.84	0.89	0.87
funct	0.90	0.90	0.84	1.00	0.93	0.91
int	0.90	0.93	0.89	0.93	1.00	0.93
overall	0.96	0.93	0.87	0.91	0.93	1.00

Table 7.9: Correlations between the six Webby scores. All of the entries are significant.

WAMMI scales are actually percentiles (e.g., a scale value of 70 means that the site scored better than 70% of Web sites and worse than 30% of sites).

- Webby scores range from 1 to 10, while WAMMI scales range from 1 to 100 (see Table 7.8); and
- All of the Webby scores are strongly correlated with each other (see Spearman correlation coefficients in Table 7.9), while WAMMI scales are not (see Table 7.6).

Despite differences between these rating schemes, they were used to directly compare judges' and participants' ratings for sites. To facilitate comparison, the WAMMI scales and Webby scores were transformed into Z scores (see Table 7.10). Each Z score is a standard deviation unit that indicates the relative position of each value within the distribution (i.e., $Z_i = \frac{x_i - \bar{x}}{\sigma}$, where x_i is the original value, $\bar{x}$ is the mean, and σ is the standard deviation). Ideally, one would start from non-normalized WAMMI scales and then compute the Z scores, but according to the WAMMI scale developers, there is no notion of non-normalized WAMMI scales. Composite WAMMI scores (wamsum) were also transformed into Z scores to compare these measures to the computed Webby factors (webbyfac). (Webby factors were already expressed as Z scores.) Z scores for the composite WAMMI scores were highly correlated with the non-normalized scales; the correlations were also significant.

Criterion	Min.	Max.	Med.
Webby Scores			
content	-2.1	1.5	0.2
nav	-2.1	1.4	0.2
vis	-2.0	1.6	0.2
funct	-1.8	1.5	0.1
int	-2.3	1.4	0.4
overall	-1.8	1.4	-0.2
WAMMI Scales			
wamattr	-1.8	3.0	-0.1
wamcon	-1.0	3.4	-0.3
wameff	-1.9	2.4	-0.2
wamhelp	-1.2	2.6	-0.4
wamlearn	-1.6	2.8	-0.3
wamglob	-1.9	3.3	-0.2

Table 7.10: Descriptive statistics for the Webby scores and WAMMI scales (Z scores). The means and standard deviations for each measure are zero and one, respectively.

Measure	Mean		Std. Dev.	
	Good	Poor	Good	Poor
exptime	**148.5**	**111.9**	93.8	83.1
ratetime	88.5	88.3	34.8	31.7

Table 7.11: Objective measures of perceived usability for good and poor sites. Bold entries represent significant differences in means.

7.6 Perceived Usability Results

The following sections summarize the relationship between participants' subjective and objective data for good and poor sites as well as the consistency between perceived usability ratings and Webby scores.

7.6.1 Perceived Usability of Good and Poor Sites (Objective and Subjective Measures)

Objective measures were analyzed to compare participants' usage of good and poor sites; how these usage patterns correlated with subjective ratings was then studied. Site exploration (exptime) and rating (ratetime) times were the only objective measures in the perceived usability condition. Table 7.11 reports results of comparing these times for sites in the good and poor category using the Mann-Whitney test. (The Mann-Whitney test [SPSS Inc. 1999] is a nonparametric alternative to t-tests for the equality of means, which is typically used for normally-distributed samples; it is related to the Wilcoxon test.) Participants explored good and poor sites for 149 and 112 seconds on average, respectively. Test results showed this difference to be significant (two-tailed p value less than 0.05); there was no significant difference for rating time.

The difference in exploration time suggests that participants spent more time exploring good sites, possibly because they were more usable or interesting. Sinha *et al.* [2001] conducted an empirical analysis of Webby scores for the 2000 Webby Awards dataset, which includes sites in this study; the authors found content to be the best predictor of Web site quality. This finding

provides some support for the hypothesis that participants may have considered good sites to be more interesting.

To test the hypothesis that participants may have considered good sites to be more usable, composite WAMMI scores (wamsum) for good and poor sites were compared. Specifically, the goal was to determine if participants rated good sites lower than poor sites, since they had spent more time on them; this result would have contradicted the hypothesis stated above. Mann-Whitney tests computed over composite ratings for good and poor sites revealed that good sites were rated higher (mean of 75.6 vs. 68.3); this difference was significant.

There were mixed results for the six WAMMI scales, most likely due to their instability; hence, results are not reported for them in this section.

7.6.2 Consistency of Perceived Usability Ratings and Webby Scores (Subjective Measures)

The previous section showed that good sites were rated more favorably on average, thus providing some evidence that there may be some consistency between judges' and participants' ratings of sites. This section explores rating consistency further by comparing mean ratings for the five Webby scores and WAMMI scales that appeared to be directly related based on their descriptions. Specifically, the analysis compares: Webby content (content) to WAMMI help (wamhelp); Webby structure and navigation (nav) to WAMMI controllability (wamcon); Webby visual design (vis) to WAMMI attractiveness (wamattr); Webby functionality to WAMMI efficiency (wameff); and Webby overall (overall) to WAMMI global usability (wamglob). The analysis also compares the Webby factor (webbyfac) to the composite WAMMI score (wamsum).

For this comparison, the WAMMI scales and Webby scores were transformed into Z scores as discussed in Section 7.5. Wilcoxon Signed Ranks tests [SPSS Inc. 1999] were then conducted on the Z scores to study mean differences between site ratings in the two schemes. (The Wilcoxon Signed Ranks test is equivalent to the paired t-test for related variables in normally-distributed samples.) If the Wilcoxon Signed Ranks test reports that a pair of ratings is significantly different, then the ratings are not consistent and vice versa. There was no difference between composite ratings – Webby factor and sum of responses to WAMMI statements; thus, perceived usability ratings were mostly consistent with judges' scores.

Judges' scores were also consistent with usability ratings for the individual scores, except for structure and navigation and controllability. The difference between the navigation and controllability measures may be due to incompatibility. Recall that mappings between Webby scores and WAMMI scales were determined strictly based on their descriptions (see Section 7.5). These two measures may actually be assessing different aspects. Furthermore, the controllability scale may be unstable due to the small number of responses per site.

7.7 Actual Usability Results

The following sections summarize the relationship between participants' subjective and objective data for good and poor sites as well as the consistency between usability ratings based on actual site usage and Webby scores.

Measure	Mean		Std. Dev.	
	Good	Poor	Good	Poor
exptime	**153.4**	**122.7**	96.1	90.5
tasktime	**196.8**	**157.8**	92.6	85.8
ratetime	89.0	92.7	30.6	33.9
t1suc	2.8	2.9	0.5	0.4
t2suc	2.8	2.9	0.5	0.4
t3suc	**2.4**	**2.6**	0.6	0.5
notry	**0.2**	**0.1**	0.5	0.2
nocomp	0.7	0.6	0.7	0.8
comp	**2.1**	**2.3**	0.9	0.9

Table 7.12: Objective measures of actual usability for good and poor sites. Bold entries represent significant differences in means.

7.7.1 Actual Usability of Good and Poor Sites (Objective and Subjective Measures)

Objective measures were analyzed to compare participants' actual usage of good and poor sites to complete information-seeking tasks; how these usage patterns correlated with subjective ratings was then studied. Objective measures in the actual usability condition include: site exploration time (exptime), task completion time (tasktime), rating time (ratetime), whether the participant did not attempt, attempted, or completed each of the three tasks (t1suc, t2suc, and t3suc), the number of tasks the participant did not attempt (notry), the number of tasks the participant did not complete (nocomp), and the number of completed tasks (comp).

Mann-Whitney tests on the perceived usability data showed that participants spent more time exploring good sites, possibly because they were more usable or interesting. Mann-Whitney tests on the actual usability data (see Table 7.12) revealed that exploration time (exptime), task completion time (tasktime), successful completion of the site-specific task (t3suc), the number of tasks not attempted (notry), and the number of tasks completed (comp) were all significantly different between good and poor sites in the full sample. Considering these measures in tandem suggests an interesting pattern – participants spent more time on good sites, but completed fewer tasks than they did on poor sites. The following significant differences supported this pattern:

- participants spent 151 seconds on average exploring good sites versus 125 seconds on poor sites;
- participants spent 198 seconds on average completing tasks on good sites versus 159 seconds on poor sites;
- t3suc measures (2.36 vs. 2.55) indicate that participants completed fewer site-specific tasks on good sites; and
- notry measures (0.18 vs. 0.05) indicate that a larger proportion of tasks (general and site-specific) were not even attempted on good sites.
- comp measures (2.1 vs. 2.3) indicate that fewer tasks (general and site-specific) were completed on good sites.

Based on this pattern, one may naturally expect that participants rated good sites lower than poor sites. However, this was not the case. Good sites had a mean wamsum score of 72.1

vs. 68.2 for poor sites, but this difference was not significant. There were mixed results for the six WAMMI scales, most likely due to their instability; hence, results are not reported for them in this section.

7.7.2 Consistency of Actual Usability Ratings and Webby Scores (Subjective Measures)

The previous section showed that good sites were rated slightly more favorably on average, although the difference was not significant. Results suggest that there may be some consistency between judges' and participants' ratings. To explore rating consistency further, the same comparison of WAMMI and Webby ratings was replicated (see Section 7.6.2). Overall, judges' scores were mostly consistent with participants' ratings. The Webby factor and composite WAMMI score were consistent as well as all other individual Webby scores, except for functionality and efficiency. There was a significant difference between the Webby functionality and the WAMMI efficiency measures, possibly due to the instability of the WAMMI scale.

7.8 Summary

This chapter presented results from a usability study of 57 Web sites wherein 30 participants rated sites in two conditions: after simply exploring sites (perceived usability); and after completing information-seeking tasks on sites (actual usability). The full data collection consisted of 550 cases. The analysis focused on answering the following questions.

- Are judges' scores consistent with perceived usability ratings?
- Are judges' scores consistent with actual usability ratings?

Analysis of objective and subjective data provided evidence that judges' scores are mostly consistent with both actual and perceived usability ratings. This suggests that profiles developed in Chapter 6 reflect usability to some degree. However, concrete conclusions about profiles reflecting usability cannot be drawn from this study due to the time difference between the judges' and users' evaluations. A follow up study needs to be conducted wherein non experts and experts rate identical sites. A better alternative is to develop the profiles from usability ratings in the first place; thus, eliminating the need for such a study. Unfortunately, the Webby Awards dataset was/is (at this time) the only large corpus of sites rated along dimensions that appear to be somewhat related to usability.

Chapter 8

Applying the Web Interface Profiles: Example Web Site Assessment

8.1 Introduction

This chapter describes the use of the profiles developed in Chapter 6 to assess and improve the quality of an example Web site. The intent of this chapter is three-fold: 1. to demonstrate how the models can be systematically applied to this problem; 2. to illustrate the type of design changes informed by the models and how they vary across models; and 3. to highlight current limitations of the models. The example assessment closely follows the evaluation scenario depicted in Chapter 4, which is the overarching goal of the work in this dissertation. Currently, interpreting model predictions and determining appropriate design changes is a manual process; future work will focus on automating recommendations for improving designs as well as implementing these recommendations. Identifying comparable good designs to aide in site improvements will also be incorporated in future work.

8.2 The Example Site

Figures 8.1–8.3 show three pages taken from a small (nine page) site in the Yahoo Education/Health category. The site provides information about training programs offered to educators, parents, and children on numerous health issues, including leukemia and cerebral palsy. The site, which was not included in the profile development sample (see Chapter 6), was selected because on first glance it appears to have good features, such as clear and sharp images and a consistent page layout, but on further inspection it seemed to have problems. The site assessment focused on answering the following questions.

- Is this a high-quality site? Why or why not?
- Are these high-quality pages? Why or why not?
- What can be done to improve the quality of this site?

The first step was to download a representative set of pages from the site. For this particular site, only eight level-one pages were accessible, and no level-two pages were reachable, for a total of 9 downloaded pages. Although there is a page containing links (Figure 8.2), the links are to pages external to the site.

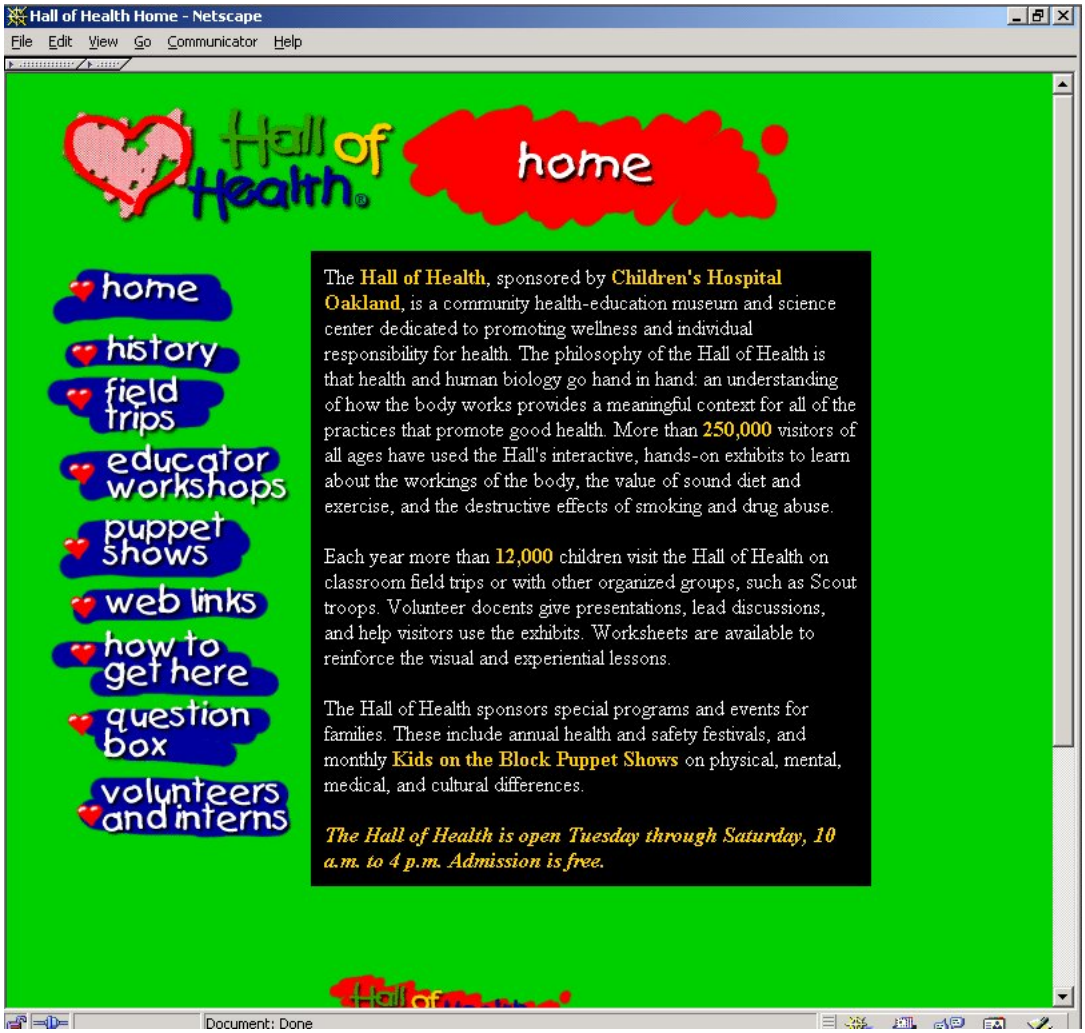


Figure 8.1: Home page taken from the example health education site (http://www.hallofhealth.org/home.html; September 14, 2001).

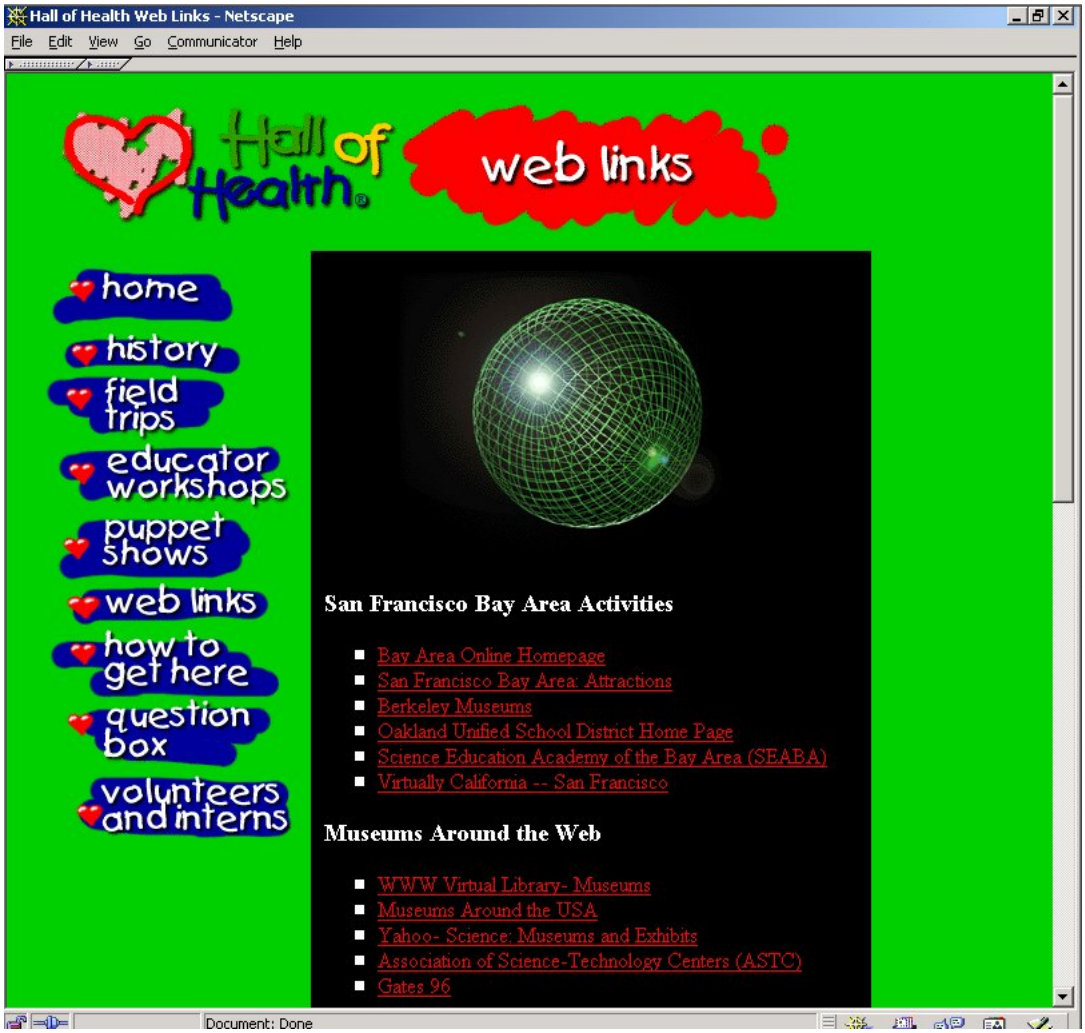


Figure 8.2: Link page taken from the example health education site (http://www.hallofhealth.org/weblinks.html; September 14, 2001).

Figure 8.3: Content page taken from the example health education site (http://www.hallofhealth.org/puppetshows.html; September 14, 2001).

The next step was to use the Analysis Tool (see Chapter 4) to compute site-level and page-level measures and to apply the models to individual pages and to the site as a whole. Each model encapsulates relationships between key predictor measures and can be used to (i) generate quality predictions and (ii) determine how pages and sites are consistent with or deviate from good pages and sites.

In the discussions below, when decision tree rules are used to generate predictions, the consequences are interpreted manually. When cluster models are applied, the score for each measure on an individual page is compared to that of the cluster centroid, and if the measure differs by more than one standard deviation unit from the centroid, the measure is reported as being inconsistent with the cluster. Cluster deviations are also interpreted manually.

8.3 Site-Level Assessment

The example site can be classified in both the health and education content categories, so the site-level decision tree model was run initially without differentiating by content category. The site-level model predicted that the site is similar to poor sites overall; the median page quality prediction (i.e., median computed over the overall page quality model's predictions for the nine pages; poor) is consistent with the overall site quality model's prediction. The corresponding decision tree rule (top of Figure 8.4) reveals that the site has an unacceptable amount of variation in link elements (31%), although variation for other site-level measures is acceptable. The combination of the link element variation and the lack of a comparable overall element variation violates patterns discovered on good sites.

The major source of link element variation is the text link count. Eight out of nine pages have from two to four text links; the remaining page has 27 text links, and acts as a links page (see Figure 8.2). The decision tree rule suggests that a link element variation level below 29% is typical on good sites. One interpretation of this finding is that good sites strive to keep the navigation structure consistent among pages and may even distribute links over multiple pages to maintain this consistency. Hence, the rule may indicate the need to similarly redistribute the links on this page.

Site quality was also assessed according to the two applicable content categories – health and education. The decision tree for health sites predicted that this is a poor health site (middle of Figure 8.4). In this case the problem is inadequate text element variation. Most of the pages on the site contain paragraphs of text without headings and use only one font face (serif); this may actually make it harder for users to scan the page to find the information they are looking for [Nielsen 2000; Spool *et al.* 1999]. The median health page quality prediction (poor) is consistent with the health site prediction.

The decision tree for education sites made a prediction contrary to that for sites overall and health sites; it found this site to be consistent with good education sites (bottom of Figure 8.4). Good health and good education sites are similar with respect to graphic formatting variation, but are quite different on the other measures, which is the cause for this disparity. However, as will be discussed below, the median education page quality is poor.

8.4 Page-Level Assessment

The decision tree model for predicting page quality reports that all nine of the pages are consistent with poor pages. The home page (Figure 8.1) contains seventeen italicized body words; pages with more than two italicized body words are considered poor pages in the model (see rule

Overall Site Quality

if ((Page Performance Variation is missing OR (Page Performance Variation $\leq$ 90.2)) AND (Overall Variation is not missing AND (Overall Variation $\leq$ 14.49)) AND (Link Element Variation is missing OR (Link Element Variation $>$ 29.195)) AND (Overall Element Variation is missing OR (Overall Element Variation $\leq$ 26.07)))

Class = Poor

This rule classifies the site as poor because the pages have acceptable page performance, overall, and overall element variation, but they have more than 29.2% variation in link elements (30.68%).

Health Site Quality

if ((Graphic Element Variation is not missing AND (Graphic Element Variation $\leq$ 32.695)) AND (Text Element Variation is missing OR (Text Element Variation $>$ 47.45)) AND (Text Element Variation is missing OR (Text Element Variation $\leq$ 92.25)))

Class = Poor

This rule classifies the site as poor because the pages have acceptable graphic element variation, but they have between 47.45% and 92.25% variation in text elements (53.18%).

Education Site Quality

if ((Median Page Breadth is missing OR (Median Page Breadth $\leq$ 11.25)) AND (Page Title Variation is missing OR (Page Title Variation $\leq$ 196.7)) AND (Page Formatting Variation is missing OR (Page Formatting Variation $\leq$ 27.785)) AND (Page Title Variation is missing OR (Page Title Variation $\leq$ 132.495)) AND (Graphic Formatting Variation is not missing AND (Graphic Formatting Variation $\leq$ 16.165)))

Class = Good

This rule classifies the site as good due to an acceptable combination of measures: the median page breadth (8) is less than twelve; and pages in the site have very little similarity in page titles (37.5%), page formatting variation (0%), and graphic formatting variation (3.19%).

Figure 8.4: Decision tree rules reported for the example health education site. The rules were reported by the overall (top), health (middle), and education (bottom) site quality models.

Home Page

if ((Italicized Body Word Count is not missing AND (Italicized Body Word Count > 2.5)))
    Class = Poor

This rule classifies the home page as poor because it contains more than two italicized words (17) in the body text.

Link and Content Page

if ((Italicized Body Word Count is missing OR (Italicized Body Word Count $\leq$ 2.5)) AND (Minimum Font Size is not missing AND (Minimum Font Size > 9.5)) AND (Minimum Graphic Height is missing OR (Minimum Graphic Height $\leq$ 36)) AND (Minimum Color Use is not missing AND (Minimum Color Use > 15.5)))
    Class = Poor

This rule classifies both the link and content pages as poor because they contain an acceptable number of italicized words in the body text and contain at least one image with a height less 37 pixels, but all of the text is formatted with a font greater than 9pt and all of the colors are used more than fifteen times. Recall that good pages tend to use a font smaller than 9pt typically for copyright text, and they use an accent color (see Section 6.6).

Figure 8.5: Decision tree rules reported for the three example pages. These rules were reported by the overall page quality model.

at the top of Figure 8.5). Schriver [1997] suggests that italicized text should be avoided because it is harder to read on computer screens than in printed documents.

Recall from Section 5.10.1 that a minimum color count metric was developed to track the number of times each color is used on a page and to report the minimum number of times a color is used; this measure detects the use of an accent or sparsely-used color. The content page (Figure 8.3) is classified as poor mainly because the minimum number of times a color is used is sixteen and all of the text, including the copyright text at the bottom of the page, is formatted with a font greater than 9pt (bottom of Figure 8.5). Good pages tend to have an accent color that they use sparingly, whereas poor pages seem to overuse accent colors (see Section 6.6). Good pages also tend to use a smaller font size for copyright or footer text unlike poor pages. Additionally, the example content page contains 34 colored body text words, which is twice the average number found on good pages; in the extreme case, a large number of colored words could result in the uncolored words standing out more so than the colored words. The same prediction and decision tree rule is reported for the link page.

To gain more insight about ways to improve page quality, each page was mapped into one of the three clusters of good pages – small-page, large-page, and formatted-page. All of the pages map into the small-page cluster and are far from the cluster centroid (median distance of 10.9 standard deviation units); the page closest to the center of this cluster has a distance of 4.0 standard deviation units (see Section 6.7). Pages in the example site deviate on key measures that distinguish pages in this cluster, including the graphic ad, text link, link text cluster, interactive object, and link word counts. Table 8.1 summarizes, for the sample content page, the ten key measures (i.e., measures that play a major role in distinguishing pages in this cluster) that deviate

Measure	Value	Cluster Range
Vertical Scrolls	2.0	(0.56–2.00)
Text Column Count	5.0	(0.62–4.36)
All Page Text Terms	129.0	(138.59–353.24)
Link Count	12.0	(12.40–41.24)
Text Link Count	2.0	(4.97–27.98)
Good Link Word Count	3.0	(7.43–49.67)
Bobby Browser Errors	6.0	(7.54–14.99)
Font Count	6.0	(3.64–5.80)
Sans Serif Word Count	0.0	(13.91–253.57)
Display Word Count	33.0	(1.13–18.67)

Table 8.1: Top ten measures that deviate from the small-page cluster for the example content page. The measures are presented in their order of importance. Each range reflects one standard deviation unit around the metric value at the cluster centroid. The page's measures are 8.33 standard deviation units from the cluster centroid.

from the cluster centroid; deviations are similar for other pages in the site. Most of these deviations, including two of the top ten measures (text link count and good link word count), can be attributed to the fact that the site provides predominately graphical links instead of text links for navigation. Table 8.1 also shows deviation on the page height (vertical scrolls), the use of words formatted with sans serif fonts (san serif word count), and the overall use of fonts (font count – combinations of a font face, size, bolding, and italics).

The quality of these pages was also evaluated using the more context-sensitive page quality models for health and education pages (as opposed to the overall model). All but two of the pages were predicted to be poor health pages, which mirrors the results of the site-level model. However, all of the pages were also predicted to be poor education pages, contrasting with the site-level model. In both cases, predictions were based on the features mentioned above. Table 8.2 summarizes the top ten measures that deviate from the two models and shows that there is some similarity between the two sets of measures, especially for measures related to text links (text link, link word, and good link word counts).

The contrast between site-level and page-level predictions demonstrate the need to incorporate page-level predictions into the site-level prediction. For example, a site can only be considered a good site if the site-level measures are consistent with good sites AND most of the pages are consistent with good pages. At the site level, the example site was highly consistent on page formatting, graphic formatting, and page performance; however, the page quality predictions show that several design aspects, such as text formatting and link elements, need to be improved. If the site-level model for education sites incorporated page-level measures, then this site would be considered a poor education site. Considering the median page quality predictions in conjunction with site quality predictions is one way to mitigate this limitation.

Finally, the quality of these pages was evaluated using the models for each page type – home, link, content, form, and other. The page type decision tree made accurate predictions for six of the nine pages, but inaccurately predicted that three pages were consistent with link pages; visual inspection suggested that these pages were actually content pages. As shown in Figure 8.6, the mispredictions were mainly due to an improper balance of link, body, and display text stemming from an overuse of image links. After correcting the page type predictions, all nine of the pages were classified as poor pages. Table 8.3 summarizes the top ten measures that deviate on

Health Page Quality			**Education Page Quality**		
Measure	Value	Model Range	Measure	Value	Model Range
Weblint Errors	0.0	(6.06–82.82)	Bobby Priority 2 Errors	3.0	(3.09–5.41)
Internal Link Count	10.0	(16.16–69.36)	Bobby Browser Errors	6.0	(9.08–20.28)
Bobby Browser Errors	6.0	(6.63–22.49)	Minimum Font Size	10.0	(8.88–9.10)
Link Count	12.0	(18.81–76.87)	Minimum Color Use	16.0	(0.20–6.22)
Redundant Link Count	1.0	(1.30–19.62)	Fixed Page Width Use	0.0	(0.42–1.20)
Graphic Pixels	224.1K	(80.6K–214.8K)	Minimum Graphic Height	32.0	(0.00–21.45)
Text Link Count	2.0	(3.41–53.91)	Text Positioning Count	14.0	(0.00–4.44)
Good Link Word Count	3.0	(6.23–98.25)	Text Link Count	2.0	(5.69–43.07)
Link Word Count	4.0	(6.58–136.78)	Link Word Count	4.0	(7.66–112.22)
Text Positioning Count	14.0	(1.03–7.11)	Good Link Word Count	3.0	(6.10–81.16)

Table 8.2: Top ten measures that deviate from the health and education page quality models for the example content page. The measures are presented in their order of importance. Each range reflects one standard deviation unit around the mean in the model.

if ((All Page Text Score is missing OR (All Page Text Score > 10.5)) AND (Good Body Word Count is not missing AND (Good Body Word Count ≤ 86.5)) AND (Link Word Count is missing OR (Link Word Count ≤ 73)) AND (Interactive Object Count is missing OR (Interactive Object Count ≤ 2.5)) AND (Good Body Word Count is missing OR (Good Body Word Count > 22.5)) AND (Good Text Color Combination is missing OR (Good Text Color Combination ≤ 5.5)) AND (Good Display Word Count is not missing AND (Good Display Word Count ≤ 0.5)) AND (Graphic Bytes is missing OR (Graphic Bytes > 38540)))
PageType = Link

This rule classifies a page as a link page because it has some similarity in content with the source page and contains few interactive objects and good text color combinations, but it also contains few link, good body, and good display (heading) words and more than 38 Kbytes for images. In other words, the page is dominated by images, image links in particular, and has inadequate text and text links.

Figure 8.6: Decision tree rule that mispredicts content pages to be link pages. This rule was returned for pages other than the three discussed in this section.

the sample content page. This model reports several deviations that were also reported by other models, including the minimum font size, minimum color use, sans serif word count, and text link count.

8.5 Summary of Assessment Findings

Tables 8.4 and 8.5 summarize the measures reported as being inconsistent for the individual pages and the site overall. Most of these measures were discussed above; however, some of the page-level measures were reported for pages other than the three example pages. Several page-level measures were reported as being inconsistent by over half of the models, including the link word and good link word counts, text link count, minimum font size, minimum color use, and Bobby browser errors. In addition, the text and link element variation measures were reported as being inconsistent at the site level.

The models provide some direct insight for resolving design issues associated with some of the measures. For example, decision tree rules reported by the overall page quality model indicate inconsistent measures with a > in the threshold (e.g., italicized body word count > 2.5); they also indicate consistent measures with a < in the threshold. In these cases, the designer could explore ways to reduce measure values below reported thresholds, such as removing italics or text coloring, changing font sizes, breaking text into multiple column, etc. The same guidance holds for the other decision tree models. Similar to decision tree rules, the cluster and discriminant classification models also provide ranges for acceptable metric values, and they report the top ten measures that deviate from the underlying models. Some of the model deviations are straightforward to correct, provided the designer understands the model output and relevant measures. Other model deviations are not as straightforward to correct, such as introducing additional links and content or reducing the reading complexity. Future work on automating design changes should make it easier to interpret and use the models to improve designs.

Based on the patterns reflected in Tables 8.4 and 8.5 and the observations generated by the analysis discussed above, a list of possible ways to improve the site was derived. The changes below are ordered based on their potential impact (how much they mitigate measures that were frequently

Measure	Value	Model Range
Minimum Font Size	10.0	(8.76–9.16)
Minimum Color Use	16.0	(0.26–6.42)
Spelling Error Count	0.0	(0.14–2.88)
Good Panel Color Combinations	1.0	(0.00–0.90)
Self Containment	2.0	(0.72–1.78)
Body Color Count	3.0	(0.80–2.68)
Average Graphic Width	207.0	(41.77–191.96)
Sans Serif Word Count	0.0	(20.18–614.98)
Minimum Graphic Height	32.0	(0.00–28.61)
Text Link Count	2.0	(2.75–37.15)

Table 8.3: Top ten measures that deviate from the content page quality model for the example content page. The measures are presented in their order of importance. Each range reflects one standard deviation unit around the mean in the model.

reported as being inconsistent). The recommendations only apply to the results generated during the initial application of the models; subsequent model applications revealed further changes that are not discussed here. No recommendations are made to address the accessibility and Weblint errors, since the roles of these measures in improving design quality are unclear. Specific changes made as well as the results of the changes are discussed in the next section.

1. Increase the number of text links and corresponding link text (text link, link word, and good link word counts). This will simultaneously increase the total number of links and internal links (link and internal link count) and decrease link element variation.

2. Use a smaller font size for some text, such as the footer text (minimum font size).

3. Decrease color overuse for page text and introduce an accent color (minimum color use).

4. Minimize or eliminate the use of italicized words in body text (italicized body word count).

5. Minimize text positioning (changes from flush left and columns where text starts; text positioning and column counts).

6. Minimize font combinations (font face, size, bolding, and italics combinations; font count).

7. Reduce the sizes of images (average graphic width, minimum graphic height, and graphic pixels).

8. Improve the page layout to reduce vertical scrolling (vertical scrolls).

9. Use tables with explicit widths to control the page layout (fixed page width use).

10. Vary the text elements and the formatting of text elements on the page (text element variation, good body and display word counts, sans serif word count).

11. Reduce the number of colors used for body text (body color count).

Measure	Page-Level								Site-Level		
			CCQ		PTQ					CCQ	
	OQ	Cls	H	E	H	L	C	Freq.	OQ	H	E
Text Element Measures											
Good Body Word Count							√	14.29%			
Display Word Count		√						14.29%			
Good Display Word Count							√	14.29%			
Link Word Count			√	√	√		√	57.14%			
Good Link Word Count		√	√	√	√			57.14%			
Spelling Error Count							√	14.29%			
Link Element Measures											
Text Link Count		√	√	√			√	57.14%			
Link Count		√	√		√			42.86%			
Internal Link Count			√		√	√		42.86%			
Redundant Link Count			√					14.29%			
Graphic Element Measures											
Graphic Ad Count					√			14.29%			
Text Formatting Measures											
Italicized Body Word Count	√				√			28.57%			
Sans Serif Word Count		√					√	28.57%			
Minimum Font Size	√			√		√	√	57.14%			
Body Color Count							√	14.29%			
Text Cluster Count						√		14.29%			
Text Column Count		√						14.29%			
Text Positioning Count			√	√				28.57%			
Link Formatting Measures											
No measures reported											
Graphic Formatting Measures											
Average Graphic Width						√	√	28.57%			
Minimum Graphic Height				√			√	28.57%			
Graphic Pixels			√					14.29%			

Table 8.4: Measures reported as being inconsistent with the page-level and site-level models for the example health education site (Table 1 of 2). A √ indicates that a measure was reported as being inconsistent on at least one of the nine pages by at least one of the models. The page-level models include the overall page quality (OQ), small-page cluster (Cls), content category quality (CCQ), and page type quality (PTQ) models. The health (H) and education (E) page models are used. The home (H), link (L), and content (C) page type models are used. The frequency column (Freq.) reflects the total number of times a measure is reported as being inconsistent divided by seven (number of page-level models). The site-level models include the overall site quality (OQ) and content category quality (CCQ) models; the health (H) and education (E) site models are used.

Measure	Page-Level								Site-Level		
			CCQ		PTQ					CCQ	
	OQ	Cls	H	E	H	L	C	Freq.	OQ	H	E
Page Formatting Measures											
Minimum Color Use	√			√		√	√	57.14%			
Good Panel Color Combinations							√	14.29%			
Bad Panel Color Combinations					√			14.29%			
Vertical Scrolls		√						14.29%			
Font Count		√				√		28.57%			
Fixed Page Width Use				√		√		28.57%			
Self Containment							√	14.29%			
Page Performance Measures											
Bobby Priority 2 Errors				√	√			28.57%			
Bobby Browser Errors		√	√	√		√		57.14%			
Weblint Errors			√		√			28.57%			
Graphic Bytes							√	14.29%			
Object Count					√			14.29%			
All Page Text Terms		√				√		28.57%			
All Page Text Score						√		14.29%			
Site Architecture Measures											
Link Element Variation									√		
Text Element Variation										√	

Table 8.5: Measures reported as being inconsistent with the page-level and site-level models for the example health education site (Table 2 of 2). A √ indicates that a measure was reported as being inconsistent on at least one of the nine pages by at least one of the models. The page-level models include the overall page quality (OQ), small-page cluster (Cls), content category quality (CCQ), and page type quality (PTQ) models. The health (H) and education (E) page models are used. The home (H), link (L), and content (C) page type models are used. The frequency column (Freq.) reflects the total number of times a measure is reported as being inconsistent divided by seven (number of page-level models). The site-level models include the overall site quality (OQ) and content category quality (CCQ) models; the health (H) and education (E) site models are used.

8.6 Improving the Site

Although the example site is somewhat aesthetically pleasing and highly consistent across pages within the site, the individual pages and the site as a whole are classified as being of poor quality. The pages were modified to incorporate a subset of the recommendations discussed above.

- To improve the color and text link counts and simultaneously reduce the link count variation, a link text cluster (i.e., an area of text links shaded with a different background color to make it stand out) was added as a footer at the bottom of each page; the text links in the cluster mirror the content of the graphical links. It was not necessary to split the link page into multiple pages, because adding the footer decreased the link element variation from 31% to 7%.

- To improve text formatting and the text element variation score: headings were added to break up paragraphs; additional font variations were used – Arial font (sans serif) for body text and Trebuchet (serif) for headings; and the font size of the copyright text was reduced to 9pt. The color of headings was also changed to gold for consistency with the models. All of these changes were implemented via an internal stylesheet; the stylesheet also improved the self-containment scores.

- To improve the emphasized (i.e., bolded, colored, italicized, etc.) body text scores, italics and colors within body text were converted to bold, uncolored body text on all pages. Colored, non-italicized body text was also converted to uncolored body text.

- To improve the minimum color usage scores, a color accent was added to the vertical bars between the text links in the footer of each page. A browser-safe color was selected as dictated by a subsequent prediction by the overall page quality model.

- To reduce vertical scrolling, the logo and copyright notice at the bottom of the pages was placed adjacent to each other in one table row. The sizes of images and borders around them were also reduced to improve space utilization. Furthermore, text was wrapped to the left of the images versus images not being inlined with text.

- To further improve the page layout, fixed widths (640 pixels) were used for the main layout table.

Figures 8.7–8.9 depict the revised pages corresponding to the pages in Figures 8.1–8.3; many of the changes are not visible since they appear at the bottom of the pages. Furthermore, only a subset of the potential changes were implemented. Appendix D provides side-by-side comparisons of the original and modified versions of the three pages.

After making these changes, all of the pages were classified correctly by functional type, and they were rated as good pages overall as well as good health pages. Figure 8.10 depicts the complex decision tree rule that classified all of the pages as good overall. The median distance to the small-page cluster was 4.7 as compared to 10.9 standard deviation units for the original pages. Eight pages were rated as average pages based on their functional type; one was rated as poor. In addition, five of the nine pages were rated as average education pages; the four remaining pages were rated as poor. These differences in predictions demonstrate the potential difficulty of satisfying all of the models simultaneously. Hence, a clear design objective needs to be chosen prior to making any changes, since the models could reveal a different set of changes to make.

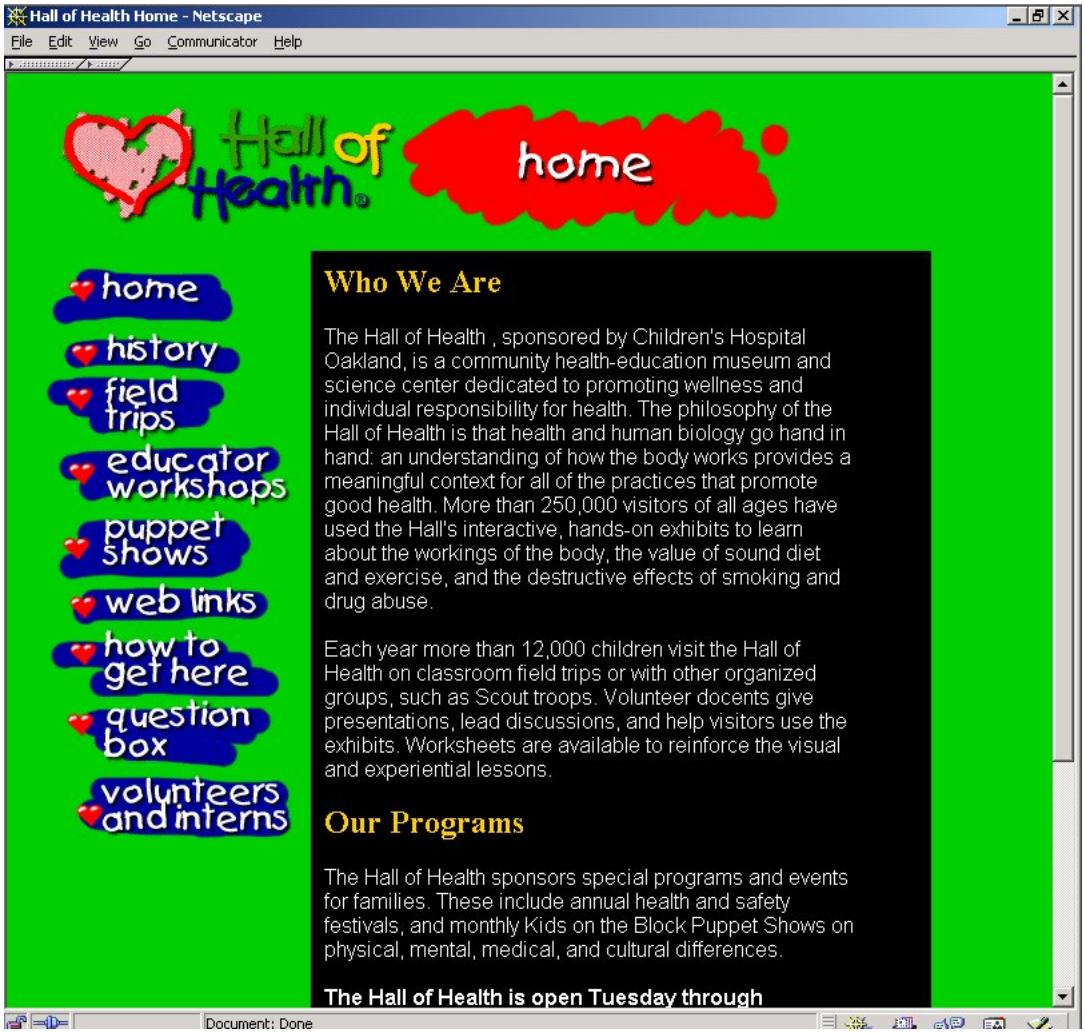


Figure 8.7: Modified home page for the example health education site. Gold headings were added, sans serif fonts were used for body text, colored and italicized body text was removed, and a fixed page width of 640 pixels was used. A footer navigation bar was added to the bottom of the page, an accent color was added to the footer navigation bar, footer elements were reorganized to reduce vertical scrolling, and the font size of footer text was reduced; none of these changes are visible in the screen shot. See Figure 8.9 for the footer navigation bar.

Figure 8.8: Revised link page for the example health education site. Gold headings were added, sans serif fonts were used for body text, colored and italicized body text was removed, the sizes of images were reduced, and a fixed page width of 640 pixels was used. A footer navigation bar was added to the bottom of the page, an accent color was added to the footer navigation bar, footer elements were reorganized to reduce vertical scrolling, and the font size of footer text was reduced; none of these changes are visible in the screen shot. See Figure 8.9 for the footer navigation bar.

Figure 8.9: Revised content page for the example health education site. Gold headings were added, sans serif fonts were used for body text, colored and italicized body text was removed, the sizes of images were reduced, and a fixed page width of 640 pixels was used. A footer navigation bar was added to the bottom of the page, an accent color was added to the footer navigation bar, footer elements were reorganized to reduce vertical scrolling, and the font size of footer text was reduced; not all of these changes are visible in the screen shot.

if ((Minimum Font Size is missing OR (Minimum Font Size $\leq$ 9.5)) AND (Graphic Ad Count is missing OR (Graphic Ad Count $\leq$ 2.5)) AND (Exclaimed Body Word Count is missing OR (Exclaimed Body Word Count $\leq$ 11.5)) AND (Minimum Graphic Height is missing OR (Minimum Graphic Height $\leq$ 38.5)) AND (Vertical Scrolls is missing OR (Vertical Scrolls $\leq$ 3.5)) AND (Bad Panel Color Combinations is missing OR (Bad Panel Color Combinations $\leq$ 2.5)) AND (Object Count is missing OR (Object Count $\leq$ 4.5)) AND (Good Meta Tag Word Count is missing OR (Good Meta Tag Word Count $\leq$ 42.5)) AND (Minimum Color Use is missing OR (Minimum Color Use $\leq$ 12.5)) AND (Horizontal Scrolls is missing OR (Horizontal Scrolls $\leq$ 0.5)) AND (Weblint Errors is missing OR (Weblint Errors $\leq$ 54.5)) AND (Colored Body Word Count is missing OR (Colored Body Word Count $>$ 0.5)) AND (Emphasized Body Word Count is missing OR (Emphasized Body Word Count $\leq$ 183)) AND (Bolded Body Word Count is missing OR (Bolded Body Word Count $\leq$ 43.5)) AND (Script Bytes is not missing AND (Script Bytes $\leq$ 173.5)) AND (Text Positioning Count is missing OR (Text Positioning Count $\leq$ 9)) AND (Serif Word Count is missing OR (Serif Word Count $\leq$ 325.5)) AND (Italicized Body Word Count is missing OR (Italicized Body Word Count $\leq$ 1.5)) AND (Graphic Count is not missing AND (Graphic Count $\leq$ 15.5)) AND (Minimum Graphic Width is missing OR (Minimum Graphic Width $\leq$ 97.5)) AND (Bobby Browser Errors is missing OR (Bobby Browser Errors $>$ 6.5)))

Class = Good

This rule classifies a page as a good page because it: uses a smaller font size for some text; has fewer than sixteen images and no graphical ads; uses at least one image with a height smaller than 39 pixels as well as at least one image with a width smaller than 98 pixels; has fewer than 183 total emphasized (i.e., italicized, bolded, colored, etc.) body words, but has fewer than 11.5 exclaimed body words (i.e., body words followed by exclamation points), fewer than 44 bolded body words, fewer than two italicized body words, and at least 1 colored body word; requires fewer than four vertical scrolls and no horizontal scrolls; starts text in nine or fewer vertical positions; uses fewer than 2.5 bad panel color combinations and uses an accent color; uses no scripts, applets, or other objects; uses fewer than 43 good meta tag words and has fewer than 325 words formatted with serif fonts; and has fewer than 55 Weblint errors and more than six Bobby browser errors.

Figure 8.10: Decision tree rule reported for all of the modified example pages. This rule was reported by the overall page quality model.

The site was still classified as a poor site overall, but for a different reason – too much text element variation. The original site had very little variation in text elements (body and display text in particular); adding headings to pages increased the text element variation (75.5%) above the acceptable threshold of 51.8%. Ensuring that all pages contain similar amounts of display text is probably the simplest way to resolve this issue. Some pages, such as the example link page, have long headings, while other pages have relatively short headings. The site was also classified as a poor health site and a good education site, consistent with classifications before the modifications; the same decision tree rules were reported (see Figure 8.4). The median overall page, education page, and health page quality predictions contradicted the site-level models.

8.7 Summary

This chapter demonstrated the ability to apply the profiles of highly-rated interfaces towards assessing and improving new sites. This capability signifies a major first step towards achieving the fundamental goal of this dissertation – enabling everyday users to produce high-quality Web sites. However, much work remains to be done to fully support this goal. In particular, an approach for suggesting interface improvements in an automated manner needs to be developed. Furthermore, an interactive evaluation tool needs to be developed to support iterative design. Future work will focus on expanding the capability demonstrated in this chapter.

The example assessment provided more insight into what the profiles actually represent and the type of design changes informed by them. The assessment suggests that the profiles provide some support for refining an implemented site, mainly improving the amount of text on the page, text formatting, color combinations, font usage, and other page layout considerations. The profiles do not support improving early site designs or the content; future work will focus on these issues.

Chapter 9

Evaluating the Web Interface Profiles

9.1 Introduction

Chapter 8 demonstrated that the profiles of highly-rated Web interfaces could be used by the author to modify an example Web site. However, it is not clear whether it is possible for others to use these profiles to modify designs. Furthermore, it is not clear whether the resulting designs are of a higher quality than the original ones.

This chapter discusses a study conducted to determine whether changes made based on two profiles – the overall page quality and the good cluster models – improve design quality. The goal of the study was to determine if participants preferred the modified pages and sites above the original ones. Whether or not preferences reflect usability was not examined in this study, and consequently is not claimed. The study also demonstrates that two undergraduate students and a graduate student were able to use the models to revise study sites.

9.2 Study Design

A study was conducted between November 26, 2001 and November 27, 2001, in accordance with guidelines established by the Committee for the Protection of Human Subjects (project 2000-1-36). Thirteen participants completed a within-subjects experiment wherein they performed two types of tasks. The first task – page-level analysis – required participants to explore two alternative designs for a Web page and to select the design that they felt exhibited the highest quality; there were a total of fifteen comparisons for pages from three sites. The second task – site-level analysis – required participants to explore a collection of pages from a Web site and to rate the quality of the site on a 5-point scale. Participants rated alternative designs for two sites; there were a total of four site ratings.

Given that only a subset of pages were modified for each site, it was not feasible to have participants attempt to complete information-seeking tasks during this study. Instead, the page-level and site-level tasks were designed to be consistent with the perceived usability condition in the usability study discussed in Chapter 7.

9.2.1 Study Sites

For the analysis, five sites were randomly selected from various Yahoo categories, such as finance and education; the sites included the one discussed in Chapter 8. Similar to the example site in Chapter 8, the other sites were selected because they had valuable content but also exhibited some

Id	#Pages	Category	Description
Site-Level Analysis			
1	9	Education Health	Information about a community health-education museum and science center as well as its teaching programs. The design consists of small pages with a few graphical links and some colored and italicized body text. This site was discussed in Chapter 8.
2	7	Education	Information on the World Wide Web and computer use for K–12 teachers. The design consists of long pages of text and text links with a logo image at the top; horizontal rules are used extensively, but very little color is used.
Page-Level Analysis			
3	5	Community	Information about a bridge club, including competition results. The design consists of small pages with tables or lists of text links, few images, and several horizontal rules.
4	5	Finance	Links to sites about the Information Economy. The design consists of very long pages with lists of annotated text links, few images, few colors, and several horizontal rules.
5	5	Living	Job listings and information about careers and employment statistics. The design consists of long pages with lots of color and images.

Table 9.1: Descriptions of sites used for the study.

problematic design issues. None of the sites were included in the statistical profile development. Table 9.1 describes the sites used for the study, and Figures 9.1–9.5 depict example pages from each site.

The Site Crawler tool was used to download pages from the five sites; the default crawling options were used (i.e., download fifteen level-one pages and three level-two pages from each level-one page). Only five pages were selected from sites 3, 4, and 5 for the page-level comparisons. All nine of the available pages were used for site 1, and seven pages were selected for site 2.

The same process followed in Chapter 8 was also followed to create modified versions of the 31 pages. Specifically, output from the overall page quality and good page cluster models was used to iteratively make changes to pages so they would be more consistent with the models. Two undergraduate students (Deep Debroy and Toni Wadjiji) and a graduate student (Wai-ling Ho-Ching) modified sites 3, 4, and 5; the author modified sites 1 and 5 and made minor final revisions to sites 4 and 5. The students had little or no training in Web design and had very little experience with building Web sites. Furthermore, they did not have prior experience with the Analysis Tool, the quantitative measures, nor the profiles.

The students made straightforward changes directly based on the decision tree rules and cluster model results. Students had to rely on their own intuition in cases were design changes were not as straightforward. The students reported that they had some difficulty making changes that were suggested by the profiles, such as increasing the number of text columns or decreasing color usage. The students also emphasized that it was not enough to use the overall page quality

Figure 9.1: Example page taken from site 1 (http://www.hallofhealth.org/puppetshows.html; September 14, 2001); this is the content page discussed in Chapter 8. The page was rated poor overall and was 8.33 standard deviation units away from the small-page cluster centroid.

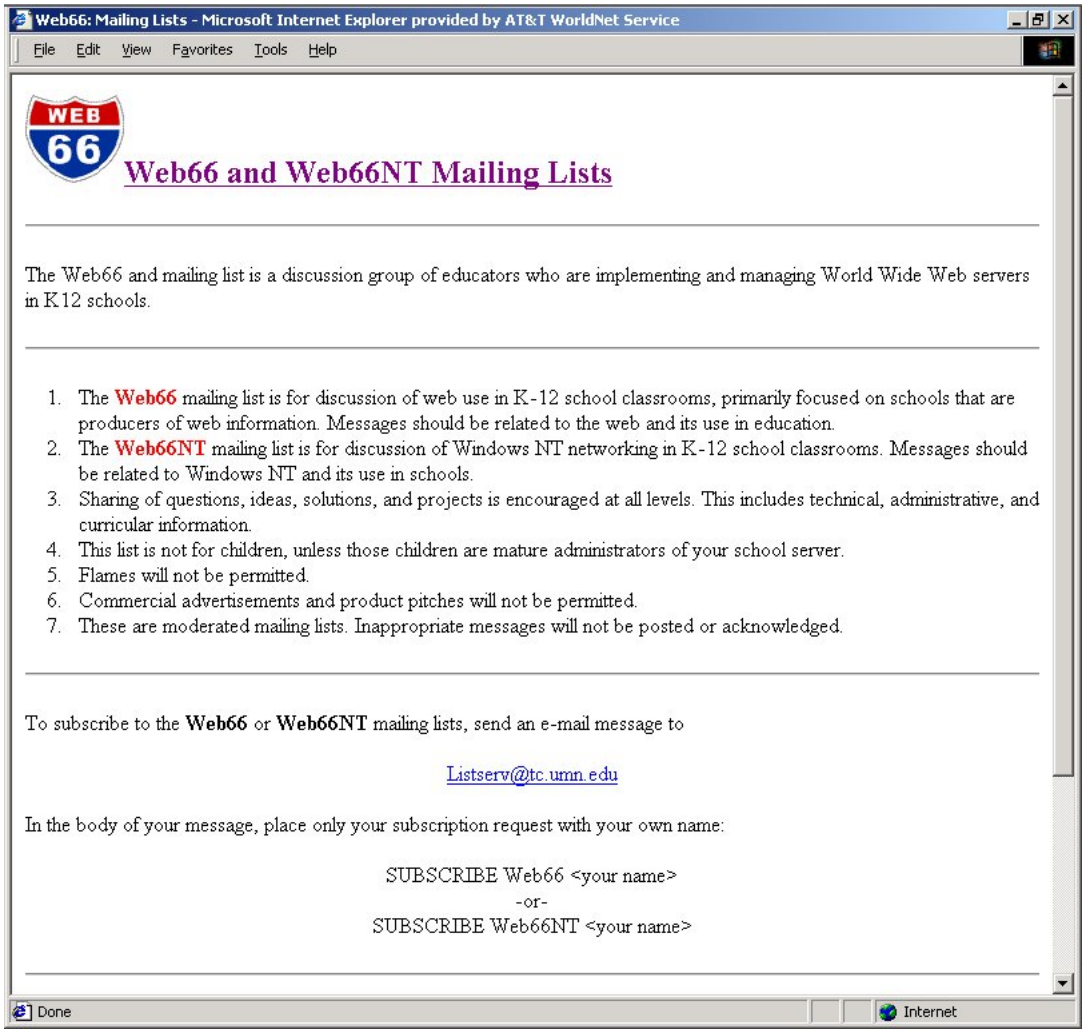


Figure 9.2: Example page taken from site 2 (http://web66.coled.umn.edu/List/Default.html; November 4, 2001). The page was rated poor overall and was 14.97 standard deviation units away from the small-page cluster centroid.

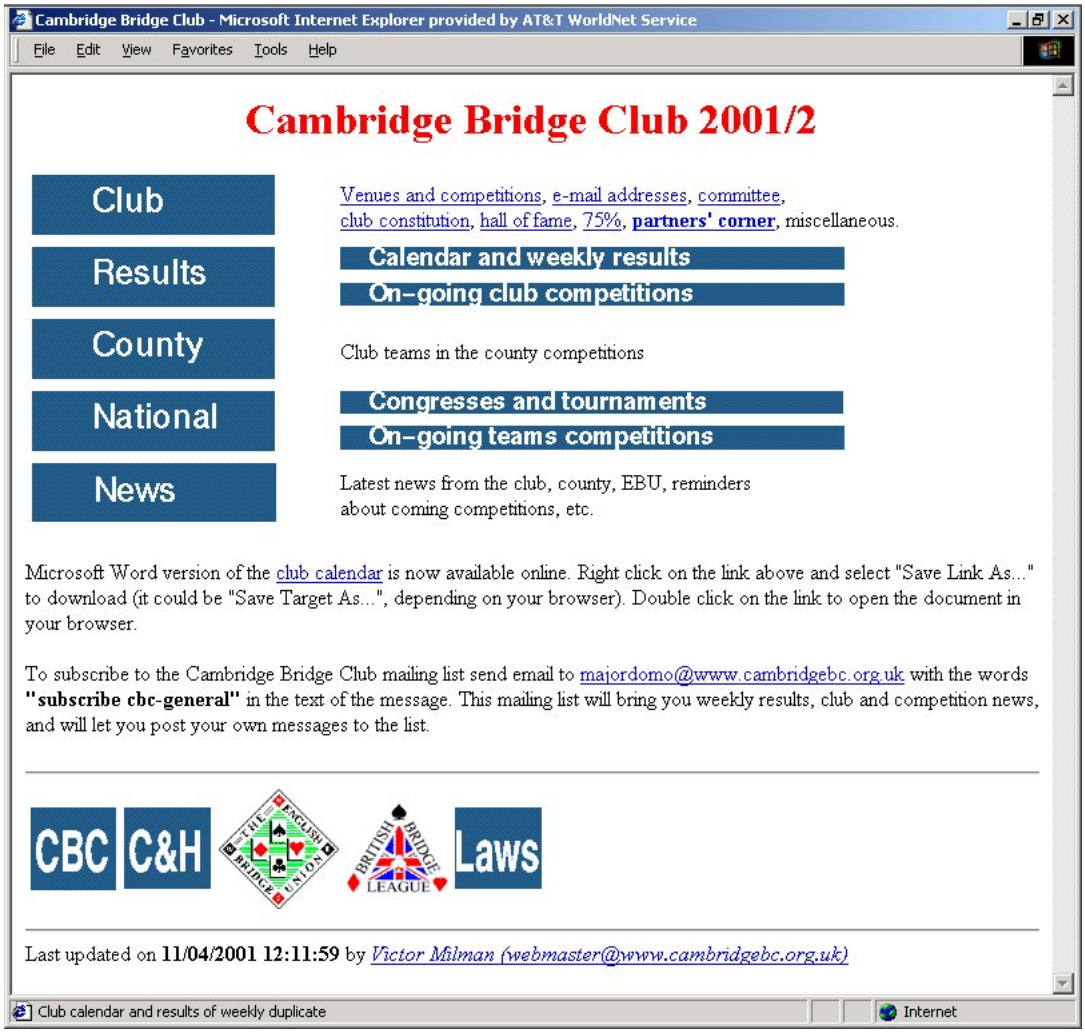


Figure 9.3: Example page taken from site 3 (http://www.cambridgebc.org.uk/CBC.html; November 4, 2001). The page was rated poor overall and was 26.46 standard deviation units away from the small-page cluster centroid.

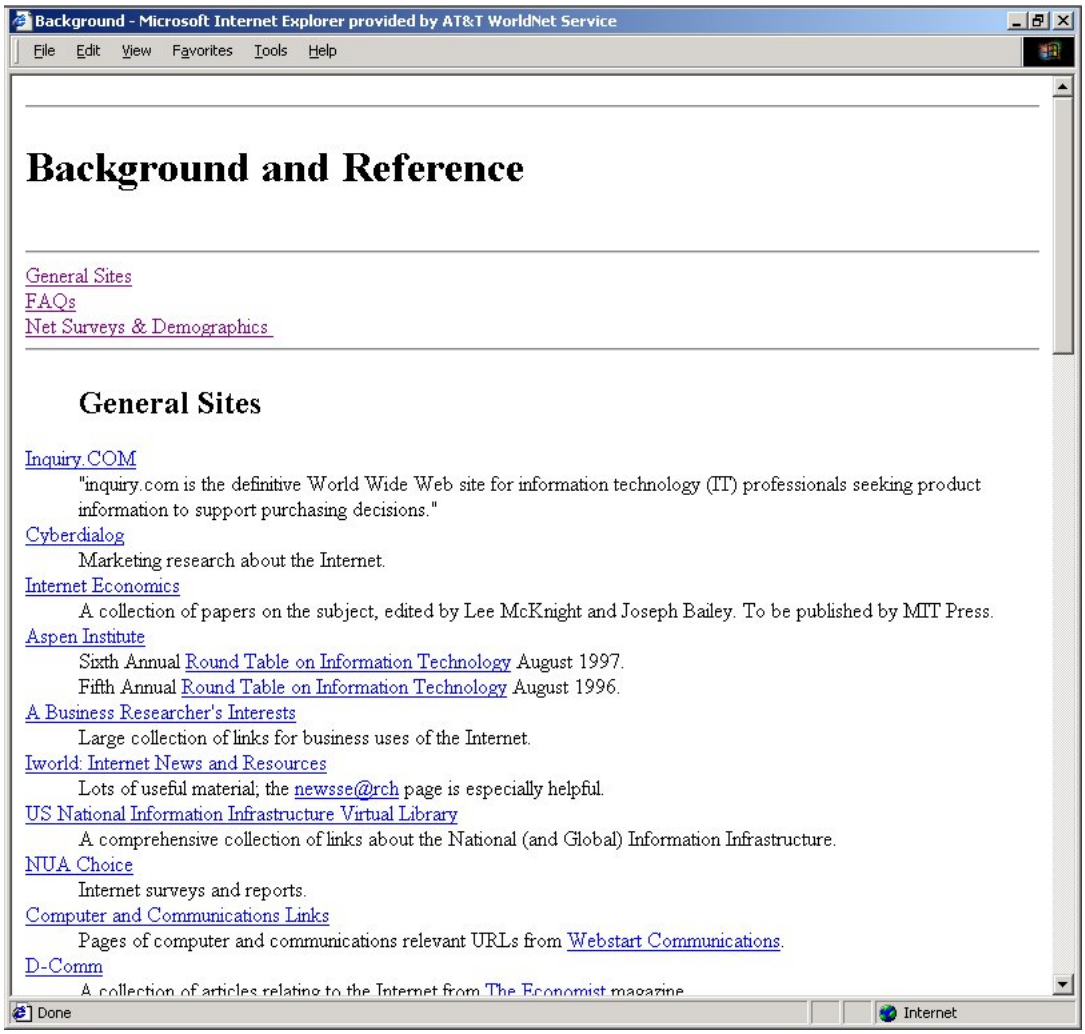


Figure 9.4: Example page taken from site 4 (http://www.sims.berkeley.edu/resources/infoecon/Background.html; November 4, 2001). The page was rated poor overall and was 32.78 standard deviation units away from the large-page cluster centroid.

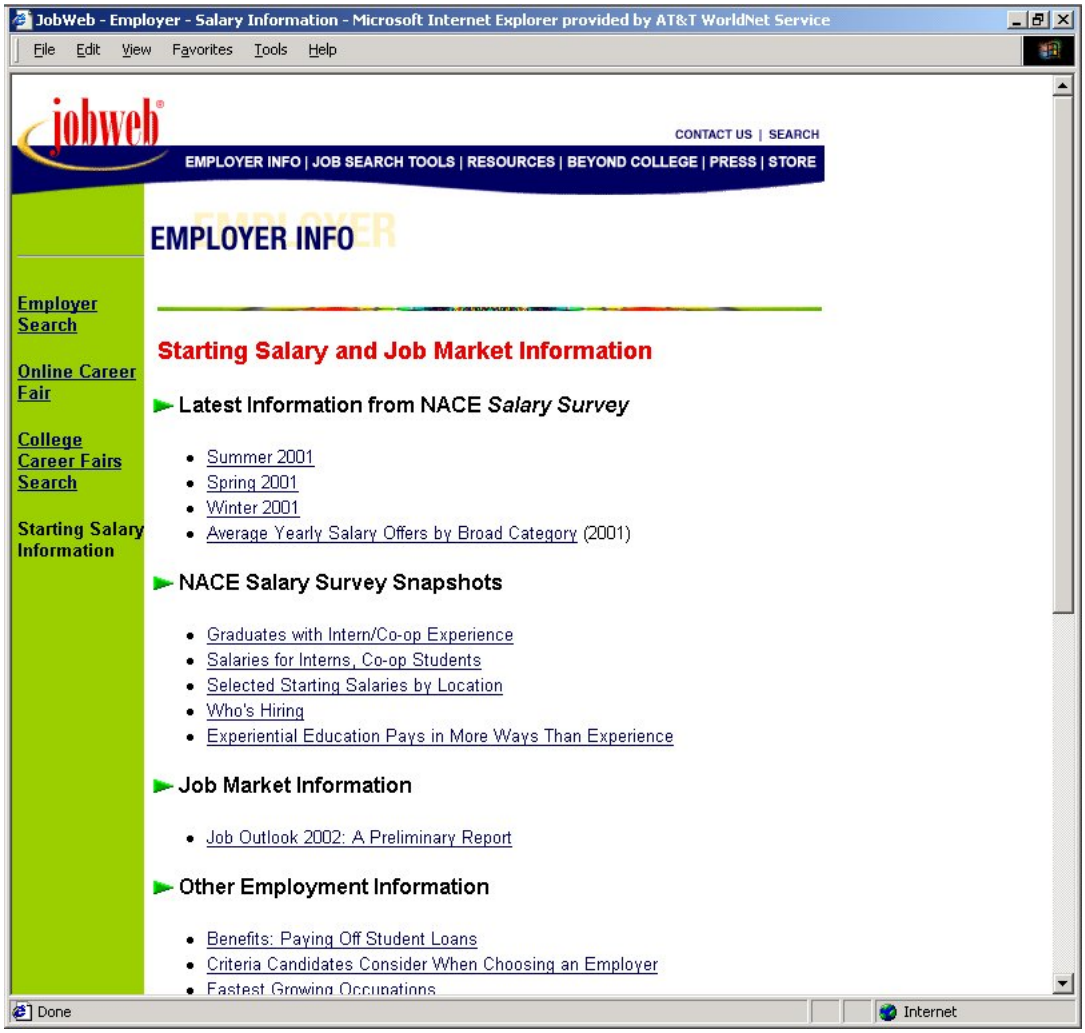


Figure 9.5: Example page taken from site 5 (http://www.jobweb.com/employ/salary/default.cfm; November 25, 2001). The page was rated poor overall and was 5.99 standard deviation units away from the small-page cluster centroid.

model by itself; the cluster models needed to be used as well. Only the overall page quality and good cluster models were used to inform design changes, and the following rules were observed.

- Changes were made solely based on the Analysis Tool results. Pages were modified to conform as much as possible to the mapped good page cluster model (subgroups of good pages with similar characteristics – small-page, large-page, or formatted-page) first; then, if possible, pages were also modified to be consistent with the overall page quality model (model that does not consider the content category or page type in predictions). The median overall page quality was used as the site quality prediction.

- The content remained the same in the original and modified sites, except in instances where footers or headings were dictated by the model results. If the models dictated that the amount of content on a page needed to be reduced, then the page was split into multiple pages as necessary.

- One change was made at a time and its impact was subsequently assessed. Changes had to result in improved or equal quality (i.e., a reduced distance from the cluster centroid or a change in prediction from poor to good) or they were discarded.

- The modified pages had to have some noticeable differences from the original pages.

Table 9.2 summarizes the differences between the original and modified pages as measured by the overall page quality and cluster models. For the original pages, 6.5% were rated good overall, 6.5% were rated average, 87.1% were rated poor, and the median distance to the mapped cluster centroid was 15.3 standard deviation units. Changes similar to those made for site 1 in Chapter 8 were also made for the four other sites. The changes included reorganizing text or images to reduce scrolling, changing text formatting (e.g., removing italicized body text), reducing text clustering, and changing colors as needed. The models revealed that other changes related to the type of content (e.g., body and link text words) and the similarity in content between source and destination pages were needed, but these changes were not made. For the modified pages, 96.8% were rated good overall, 3.2% were rated average, and the median distance to the mapped cluster centroid was 6.31 standard deviation units. The predictions suggested that there were potentially noticeable differences between pages in most cases. Figures 9.6–9.10 depict modified versions of the example pages in Figures 9.1–9.5. Appendix D provides side-by-side comparisons of the original and modified versions of the five pages.

9.2.2 Participants

Study participants were recruited from Kaiser Permanente's Web Portal Services Group[1]. This group is responsible for designing, building, and maintaining numerous intranet and Internet sites; hence, designing quality Web interfaces is extremely important to people within the group. Thirteen participants completed the study; participants represented the three roles below.

- **Professional Web Designers** - have received formal training (i.e., earned a college or art school degree) in Web or graphic design and have actively designed Web sites. These participants were employed as designers; four of the thirteen participants were from this group.

[1]The author was employed as a member of this group at the time of the study and had working relationships with some of the study participants. Participants were not informed about the purpose of the study or the hypotheses being tested.

Pair Id	Original Page			Modified Page		
	Quality	Cluster	Distance	Quality	Cluster	Distance
Site 1						
–	Poor	Small-Page	22.74	Good	Small-Page	5.16
–	Poor	Small-Page	16.5	Good	Small-Page	4.61
–	Poor	Small-Page	18.74	Good	Small-Page	17.68
–	Poor	Small-Page	6.81	Good	Small-Page	3.88
–	Poor	Small-Page	6.26	Good	Small-Page	3.81
–	Poor	Small-Page	7.95	Good	Small-Page	5.11
–	Poor	Small-Page	10.88	Good	Small-Page	4.56
–	Poor	Small-Page	8.33	Good	Small-Page	5.04
–	Poor	Small-Page	15.94	Good	Small-Page	4.72
Site 2						
–	Poor	Small-Page	21.05	Good	Small-Page	10.86
–	Poor	Small-Page	14.97	Good	Small-Page	7.83
–	Poor	Small-Page	14.68	Good	Small-Page	6.31
–	Poor	Small-Page	20.42	Good	Small-Page	6.83
–	Poor	Small-Page	15.03	Good	Small-Page	6.58
–	Poor	Large-Page	18.14	Good	Large-Page	19.2
–	Poor	Small-Page	16.21	Good	Small-Page	7.59
Site 3						
1	Poor	Small-Page	26.46	Good	Small-Page	5.86
2	Average	Small-Page	14.6	Good	Small-Page	5.71
3	Poor	Small-Page	14.77	Good	Small-Page	6.05
4	Poor	Small-Page	15.3	Good	Small-Page	7.32
5	Poor	Small-Page	15.09	Good	Small-Page	7.52
Site 4						
6	Poor	Large-Page	32.78	Good	Small-Page	11.93
7	Poor	Large-Page	23.05	Good	Small-Page	394.19
8	Poor	Large-Page	49.3	Good	Small-Page	394.18
9	Good	Large-Page	38.61	Good	Large-Page	11.79
10	Poor	Small-Page	8.31	Good	Small-Page	7.19
Site 5						
11	Average	Small-Page	15.43	Average	Small-Page	15.48
12	Poor	Small-Page	11.23	Good	Small-Page	4.82
13	Poor	Large-Page	93.87	Good	Small-Page	4.39
14	Poor	Small-Page	5.56	Good	Small-Page	5.42
15	Good	Small-Page	5.99	Good	Small-Page	5.79

Table 9.2: Model predictions for the original and modified pages. The numbers in the first column are for the page-level analysis; the – indicates pages that are included in the site-level analysis. The quality predictions are from the overall page quality model. The reported clusters and cluster distances are from the good page cluster models; the distance reflects the number of standard deviation units of difference between metric values on a page and metric values at the centroid of a cluster. In most cases, the modified pages are closer to cluster centroids and are predicted to be of a higher quality than the original pages.

Figure 9.6: Modified page for site 1; this is the modified content page discussed in Chapter 8. The page was rated good overall and was 5.04 standard deviation units away from the small-page cluster centroid. The original page was rated poor overall and was 8.33 standard deviation units away from the small-page cluster centroid (see Figure 9.1).

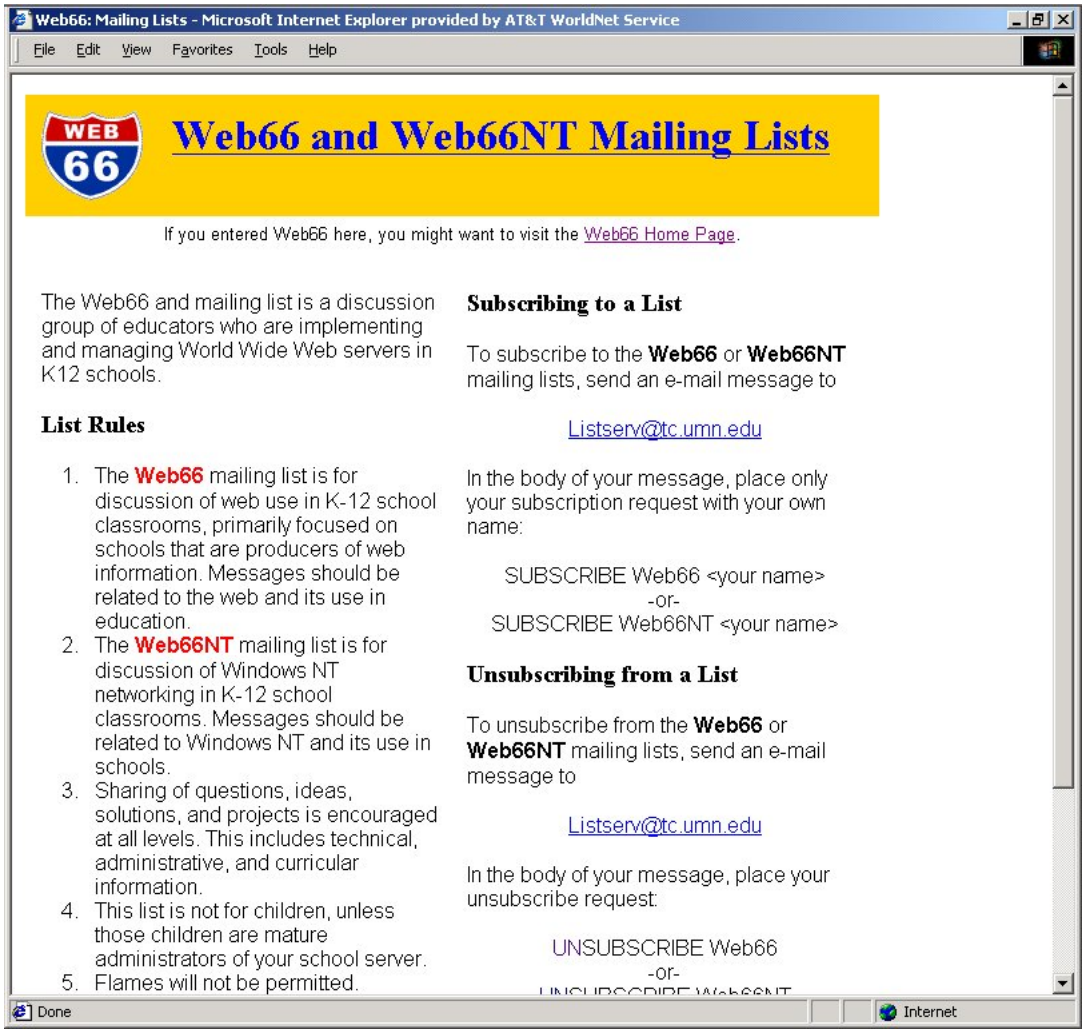


Figure 9.7: Modified page for site 2. The page was rated good overall and was 7.83 standard deviation units away from the small-page cluster centroid. The original page was rated poor overall and was 14.97 standard deviation units away from the small-page cluster centroid (see Figure 9.2).

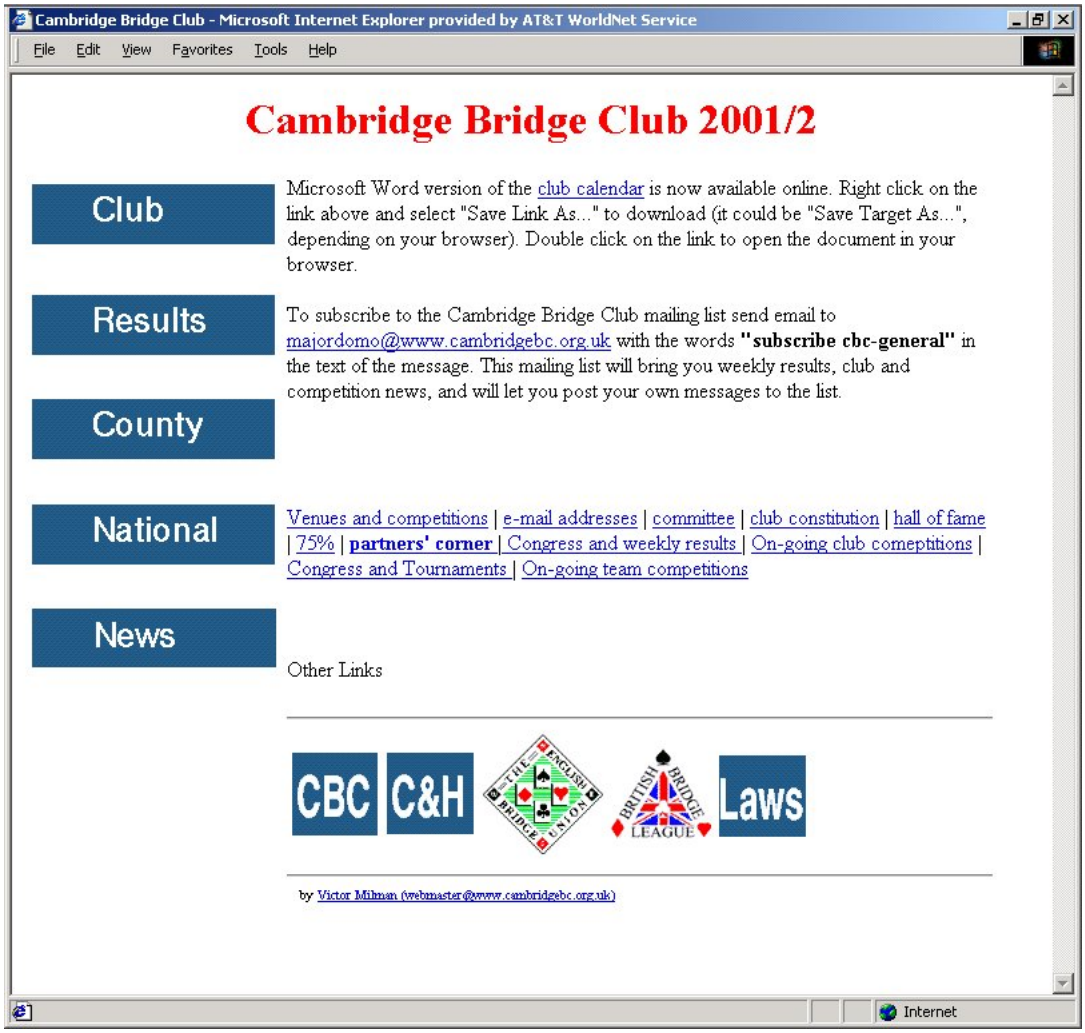


Figure 9.8: Modified page for site 3. The page was rated good overall and was 5.86 standard deviation units away from the small-page cluster centroid. The original page was rated poor overall and was 26.46 standard deviation units away from the small-page cluster centroid (see Figure 9.3).

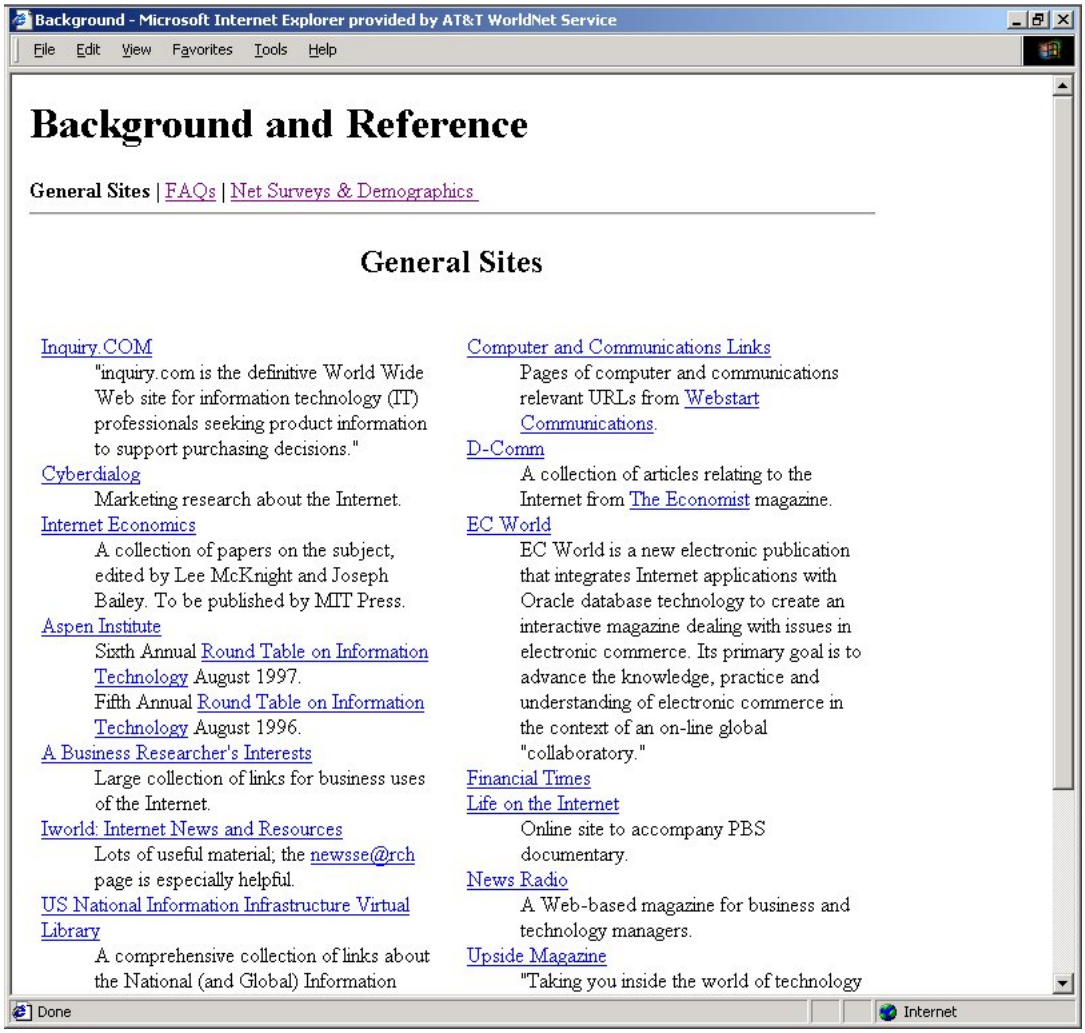


Figure 9.9: Modified page for site 4. The page was rated good overall and was 11.93 standard deviation units away from the small-page cluster centroid. The original page was rated poor overall and was 32.78 standard deviation units away from the large-page cluster centroid (see Figure 9.4).

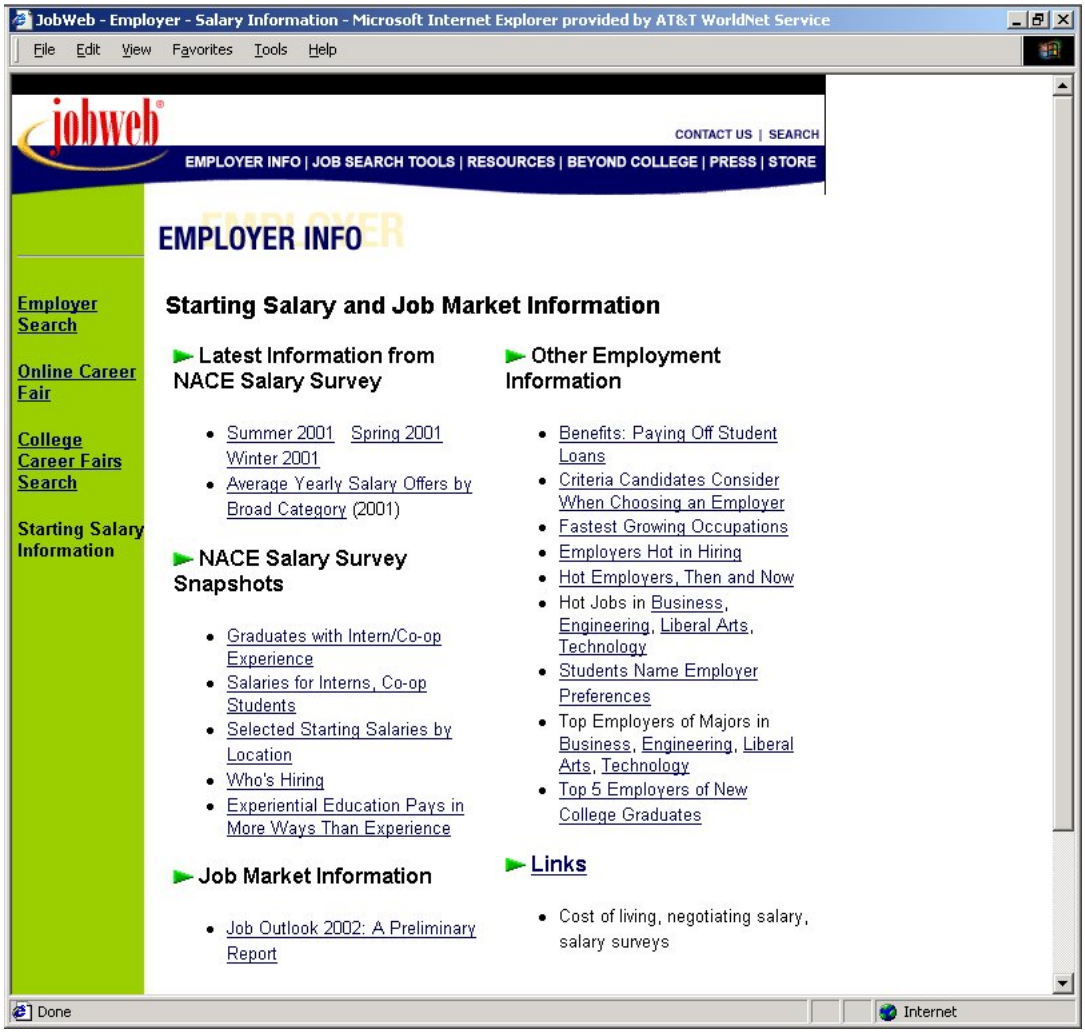


Figure 9.10: Modified page for site 5. The page was rated good overall and was 5.79 standard deviation units away from the small-page cluster centroid. The original page was rated poor overall and was 5.99 standard deviation units away from the small-page cluster centroid (see Figure 9.5).

Id	Age	Gen	Education	CmpExp	IExp	IUse	EngExp	VI
1	46-55	Male	Some College	Beginner	Average	1–2	Average	No
2	26-35	Male	College Graduate	Expert	Expert	>10	Expert	No
3	26-35	Female	College Graduate	Average	Average	6–10	Expert	No
4	36-45	Female	Post Graduate	Average	Expert	>10	Expert	No
5	>55	Female	College Graduate	Expert	Expert	1–2	Expert	No
6	26-35	Female	College Graduate	Average	Average	>10	Average	No
7	36-45	Male	Some College	Expert	Expert	>10	Average	No
8	46-55	Male	Post Graduate	Average	Expert	>10	Expert	No
9	36-45	Male	Post Graduate	Expert	Expert	>10	Expert	No
10	26-35	Female	College Graduate	Expert	Expert	6–10	Expert	No
11	26-35	Male	College Graduate	Expert	Expert	>10	Expert	No
12	46-55	Female	Post Graduate	Expert	Expert	>10	Expert	No
13	26-35	Female	College Graduate	Average	Expert	3–5	Expert	No

Table 9.3: Summary of participants' demographic information. Participants provided their age and gender (Gen) and described their proficiency with computers (CmpExp), the Internet (IExp), and the English language (EngExp). Participants also reported the number of hours they spend using the Internet weekly (IUse) and whether or not they had a visual impairment that possibly interfered with their ability to assess Web design quality (VI).

- **Non-Professional Web Designers** - have not received formal training (i.e., no degree) in Web or graphic design, but have played a role in designing Web sites or creating Web pages. These participants were employed as Web Coordinators; three of the thirteen participants were from this group.
- **Non Web Designers** - have not received formal training, have not designed Web sites, and have not created Web pages. These participants were typically developers and managers; six of the thirteen participants were from this group.

Participants answered demographic questions prior to starting the study. Specifically, participants provided their age, gender, as well as information about their education background, computer and Internet experience, the number of hours they use the Internet weekly, English experience, and whether they had a visual impairment that could interfere with their ability to assess the quality of Web interfaces. Table 9.3 summarizes responses to these questions. Participants were also asked several questions about their role and experience with designing Web sites, including the number of sites they have designed. Table 9.4 summarizes responses to these questions.

There were seven female and six male participants. The typical participant was 26–35, a college graduate, an expert computer and Internet user, spent more than ten hours online weekly, was an expert with the English language, and had no visual impairments. None of the participants had visual impairments. All of the professional Web designers have designed more than ten sites; half of them felt that they were experts at creating quality sites, while the other half felt that they were average. Two of the non-professional Web designers have also designed more than 10 sites and all of them felt that they were experts at creating quality sites. Most of the non Web designers had designed from zero to three sites and felt that they were beginners at creating quality sites. Based on the demographic information, all of the participants appear to have been appropriate for this study.

Id	Role	DesignExp	#Sites Designed
1	Non Web Designer	Beginner	0–3
2	Professional Web Designer	Expert	>10
3	Non Web Designer	Beginner	0–3
4	Non Web Designer	Average	0–3
5	Non-Professional Web Designer	Expert	0–3
6	Non Web Designer	Beginner	0–3
7	Professional Web Designer	Average	>10
8	Non Web Designer	Beginner	0–3
9	Non-Professional Web Designer	Expert	>10
10	Professional Web Designer	Expert	>10
11	Non-Professional Web Designer	Expert	>10
12	Non Web Designer	Average	4–10
13	Professional Web Designer	Average	>10

Table 9.4: Summary of participants' Web design roles and experience. Participants described their proficiency with creating quality Web sites (DesignExp) and reported the number of sites they had designed prior to the study.

9.2.3 Testing Interface

A testing interface was developed using HTML forms, JavaScript, and PERL. In addition, a script was developed to generate five randomized experiment designs. The order of page comparisons was randomized and controlled such that pages from the same site were not compared consecutively; the presentation of the two page designs was also randomized. Similarly, the order of site ratings was randomized and controlled such that the two versions of a site were not rated consecutively.

9.2.4 Testing Session

The study consisted of a 1-hour session wherein participants completed 15 page-level comparisons and four site-level evaluations. Participants were initially given an instruction sheet (see Figure 9.11) that provided an overview of the study procedure. After reviewing the instruction sheet, participants completed a statement of informed consent and moved to a computer station for completing the study.

The study interface requested demographic information and assigned the participant to one of the 5 experiment designs. The interface subsequently guided participants through two types of tasks as discussed below.

- **Page-Level Analysis.** Participants were instructed to initially explore alternative versions of a Web page. Then, participants were asked to select the version that they felt exhibited the best quality. Participants were also asked to explain why they selected the design. Figure 9.12 depicts the screens for performing the page-level tasks.

- **Site-Level Analysis.** Participants were instructed to initially explore the pages from a Web site. Then, participants were asked to rate the quality of the site using a 5-point Likert scale; ratings ranged from very poor (1) to very good (5). Participants were also asked to explain why they rated the site as they did. Figure 9.13 depicts the screens for performing the site-level tasks.

Web Site Quality Study Instructions

I. Overview of the Study
The purpose of this study is to get user feedback on the quality of a collection of web pages and sites. Basically, you will explore the pages and sites and provide responses to a number of statements about their quality. There is no risk to you, and you will be compensated with a free lunch for your participation.

Please read and sign the informed consent form and return to the tester.

II. Overview of the Study Procedure

General goal:
Rate web pages and sites. There will be two kinds of tasks. (a) Compare two alternative designs for a web page (b) Explore pages on a site and rate the site.

Tasks
(a) **Compare two alternative designs for a web page.**
In the first part of the task, you will be asked to explore the two designs. Look at each design and select the design that you feel exhibits the best quality. **Please do not spend more than a few minutes exploring the designs.**
You are encouraged to comment on why you selected one design over the other one.

(b) **Explore pages on a site and rate the site.**
You will be presented with web pages from a site and asked to explore the pages. Then rate the site. **Please do not spend more than a few minutes exploring the pages.**

Rating the site:
You will be asked to rate the site on a 5-point scale (Very Poor - Very Good). You are encouraged to comment on why you rated the site as you did.

Figure 9.11: Instructions given to study participants.

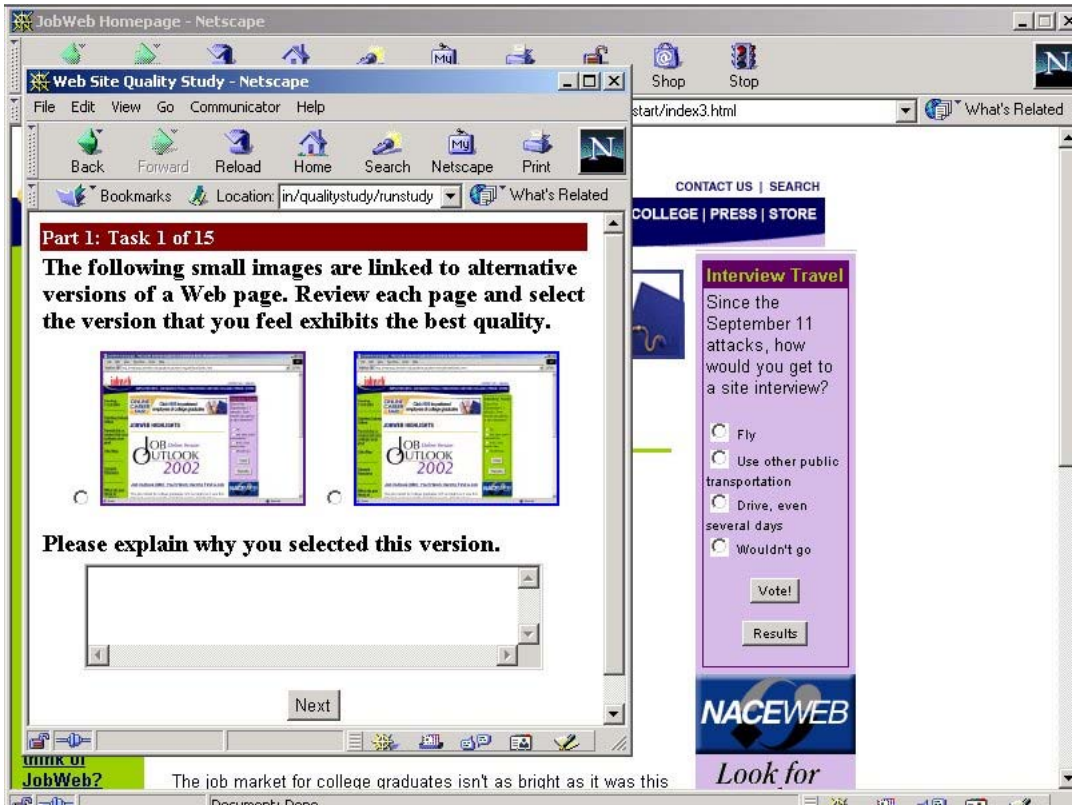


Figure 9.12: Testing interface for completing the page-level analysis tasks. The smaller browser window provides links to pages that are displayed in the larger window.

During the testing session, participants completed the page-level tasks and then completed the site-level tasks. The testing interface tracked whether participants explored all of the pages in both sections of the study. A message window appeared whenever participants did not explore all of the pages; participants were also restricted from moving forward in the study until they explored all pages. All of the links on pages were redirected to an error page reminding participants to stay focused on the individual pages.

9.2.5 Testing Environment

Participants completed the study in either the Kaiser Permanente Web Portal Services Group's computer training room or at their office computer; participants worked individually in both scenarios. All computer stations had PCs running Microsoft Windows NT 4.0 with 128 MB of RAM. Stations also had 15" or 17" monitors and high speed LAN connections. Participants used the Netscape Navigator 4.7 browser. The testing interface resized the browser window to 800 x 600 pixels (equivalent to a 15" monitor) and sites were not cached to provide a realistic experience. User surveys have shown that over 50% of Internet users access sites with 800 x 600 monitor resolution and 56K and slower connection speeds [DreamInk 2000].

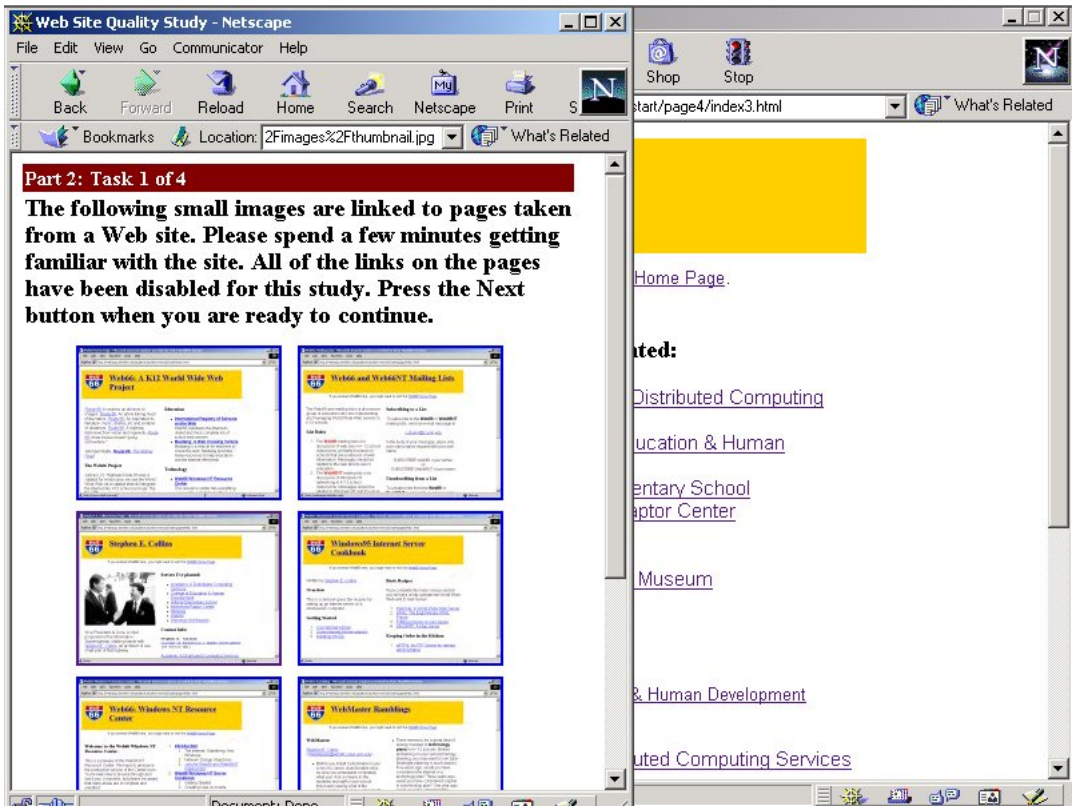


Figure 9.13: Testing interface for completing the site-level analysis tasks. The smaller browser window provides links to pages that are displayed in the larger window.

9.3 Data Collection

The analysis data consisted of 195 page-level (i.e., page preference) responses and 26 site-level (i.e., ratings of the original and corresponding modified sites) responses. Participants provided freeform comments in most cases. The site-level ratings were screened to replace small outliers with the next largest value as appropriate; the resulting data followed a normal distribution and exhibited equal variances.

9.4 Page-Level Results

The page-level analysis focused on testing the hypothesis that pages modified based on the overall page quality and the good cluster models are of a higher quality than the original pages. Recall that remodeling did not entail changing the content on pages, except for redistributing content over multiple pages, adding headings, etc. as dictated by the models. Thus, most of the differences between the original and modified pages were minimal and focused on the page layout and text formatting. The results showed that modified pages were preferred 57.4% of the time, while the original pages were preferred 42.6% of the time. The Chi-Square test revealed this difference to be significant ($\chi^2 = 4.3$, asymptotic significance of .038). ANOVAs revealed that there were no differences in preferences among the three Web design roles.

Figure 9.14 depicts results for the fifteen page-level comparisons. Participants preferred the modified pages in ten of the fifteen comparisons. Their comments about why they preferred the modified pages supported the changes made based on the profiles. In particular, participants felt that the modified pages were easier to read, required less scrolling, were cleaner, used better color schemes, made better use of whitespace, used headings, eliminated italics, and used better fonts. Several comments are provided below.

> "2 columns with shorter lines of te[x]t [is] easier to read. Primary navigation under the headline gives i[t] more prominence. Overall composition is better." (page pair 6, professional Web designer)

> "all page fits in window." (page pair 2, non-professional Web designer)

> "The text is easy to read. The text on the other page is to[o] long." (page pair 9, non Web designer)

Comments also revealed that in some cases, mainly with page pairs 1, 3, 4, 5, and 11, participants responded negatively to changes made based on the profiles. For example, modifications for the first page pair (site 3) included reorganizing graphical links and texts to minimize vertical scrolling. However, participants preferred the original version of this page because the width of text was restricted to fit within the browser window (i.e., the page did not require horizontal scrolling). This horizontal scrolling was introduced when the undergraduate student modified the page; this was an unfortunate mistake that arose because of the student's inexperience with HTML and Web design. This change did not impact the overall quality prediction, nor was it among the top ten deviations from the cluster model; most of the top ten deviations were associated with text and text formatting measures, such as the sans serif word count and text cluster and column counts.

Figure 9.14 shows that for four of the five pages on site 3 (page pairs 1–5), the original pages were preferred over the modified pages. As the Chi-Square test showed, this was not an expected outcome. Examining the model output for the modified pages revealed that other changes, such

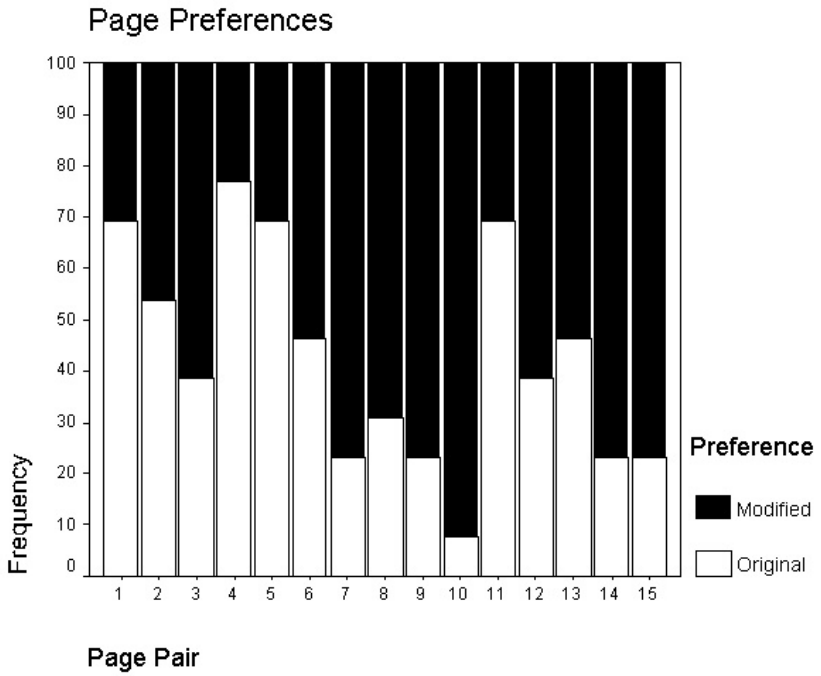


Figure 9.14: Results for the 15 page-level comparisons. Comparisons 1–5 were for pages from site 3. Comparisons 6–10 were for pages from site 4. Comparisons 11–15 were for pages from site 5.

as using sans serif fonts for text, were not implemented by the undergraduate student. Hence, it appears that the modified pages were insufficiently remodeled for comparison. If responses for pages on this site are excluded, then the results for the 2 remaining sites show that modified pages were preferred 66.9% of the time, while the original pages were preferred 33.1% of the time; this difference was highly significant ($\chi^2 = 14.9$, asymptotic significance of .000).

Several participants reported that in some cases, they could not tell the difference between the two pages and selected the first one; pair thirteen is one example. Given the restrictions on page modifications, it was not always possible to produce radically different designs. However, several participants also verbally commented on how subtle changes made a lot of difference.

9.5 Site-Level Results

The site-level analysis focused on testing the hypothesis that sites with pages modified based on the overall page quality and the good page cluster models are of a higher quality than the original sites. Similarly to the page-level analysis, the profiles were used to modify individual pages in the site. The overall site quality model was not used to assess the site given the discrepancies discussed in Section 8.6. Instead, the median overall page quality was used for site level assessments; as the quality of the individual pages improved, so did the median overall page quality. Table 9.5 showed that almost all of the original pages were rated poor overall, while almost all of the modified pages were rated good overall; the corresponding median overall page quality was poor and good, respectively.

Figures 9.15 and 9.16 depict the distributions of ratings for the original and modified sites; both distributions are nearly normal. Participants rated the quality of the original sites as 3.0 on average ($\sigma = 1.36$); however, they rated the quality of modified sites as 3.5 on average ($\sigma = 1.03$). A paired samples t-test revealed that this difference was significant ($p = .025$); this means that

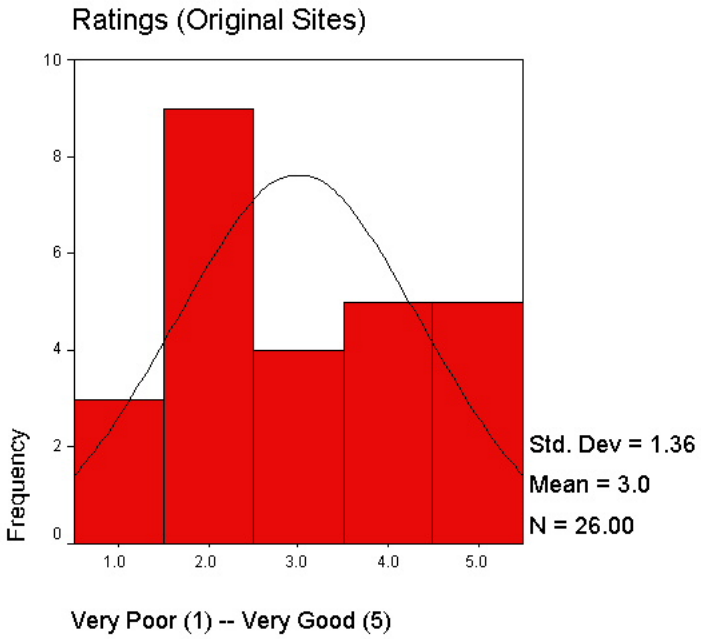


Figure 9.15: Distribution of ratings for the original versions of the 2 sites.

	Mean		Std. Dev.		
Design Role	Orig.	Mod.	Orig.	Mod.	Sig.
Professional Web Designer	2.63	3.38	1.30	0.92	0.020
Non-professional Web Designer	3.00	3.83	1.26	0.98	0.042
Non Web Designer	3.25	3.42	1.48	1.16	0.674

Table 9.5: Site ratings for the three groups of participants: 4 professional Web designers; 3 non-professional Web designers; and 6 non Web designers. Paired t-tests were used to compute the significance values.

each participant tended to rate the modified version higher than the original version. Similarly to the page-level analysis, participant comments provided support for many of the changes made based on the profiles.

There were differences in ratings among participants in the three roles. Table 9.5 shows a wider difference in mean ratings for the professional and non-professional Web designers; these differences are also significant. However, the table shows little difference in average ratings for the non Web designers, and the difference is not significant. Thus, it appears that the site level results are somewhat skewed possibly by the non Web designers' inability to gauge differences between the two versions of the sites. The comments showed that they often questioned whether they were rating the same site again. Some of these participants even stated that they rated both versions of the site the same.

For the analysis above, ratings for both sites were aggregated; however, participants' ratings for the individual sites were also examined. The modified versions of both sites were rated higher than the original versions, but the differences were not significant in most cases (see Table 9.6). Similarly to the discussion above, there were differences in ratings among the three participant groups. With respect to the example site discussed in Chapter 8 (site 1), Table 9.6 shows that in all cases participants rated the modified version slightly higher than the original site, although the

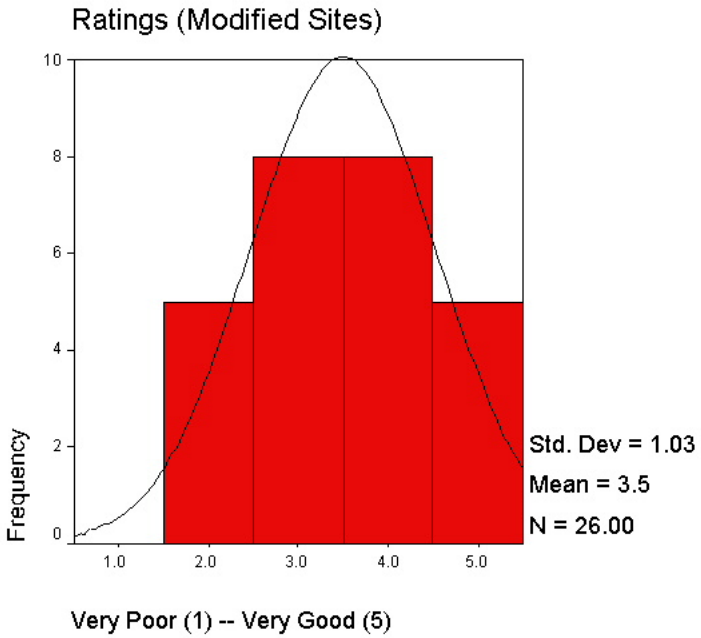


Figure 9.16: Distribution of ratings for the modified versions of the 2 sites.

difference was not significant in any case.

The intent of site remodeling was not to produce the best possible site and no claim is made about the modified sites exhibiting high quality. Due to the restrictions on page modifications (see Section 9.2.1), there were many issues that were not addressed, such as changing color schemes and adding images and links. Some of these issues were reported by the models and others, such as illegible text in images, were not. In many cases, participants' ratings and comments reflected their negative responses to these outstanding issues; it is likely that their responses also skewed the results. For example, Table 9.6 shows that professional Web designers rated sites lower than non-professional and non Web designers even though they rated the modified sites higher. The intent of site remodeling was to improve the quality of the designs in spite of the inherent problems; the results suggest that this goal was accomplished.

9.6 Summary

This chapter presented results from a user study of five Web sites wherein thirteen participants (four professional, three non-professional, and six non Web designers) completed two types of tasks: 1. explore alternative versions of Web pages and select the ones exhibiting the highest quality; and 2. explore pages from sites and rate the quality of the site. In preparation for the study, three students – two undergraduates and one graduate – used the overall page quality and good page cluster models to modify pages. The students were given access to an interactive appendix that summarized all of the quantitative measures and were able to ask the author questions about the models and measures as needed. The study demonstrated that it was possible for people other than the author to interpret and apply the models. The analysis focused on answering the following questions.

- Do participants prefer the modified pages over the original pages?
- Do participants rate the modified sites higher than the original sites?

Site	Mean		Std. Dev.		Sig.
	Orig.	Mod.	Orig.	Mod.	
All Roles					
1	3.62	3.92	1.26	1.04	0.219
2	2.38	3.08	1.19	0.86	0.069
Professional Web Designers					
1	3.50	3.75	1.29	1.26	0.391
2	1.75	3.00	0.50	0.00	0.015
Non-professional Web Designers					
1	4.00	4.67	1.00	0.58	0.423
2	2.00	3.00	0.00	0.00	–
Non Web Designers					
1	3.50	3.67	1.52	1.03	0.695
2	3.00	3.17	1.55	1.33	0.822

Table 9.6: Ratings for the 2 study sites. Paired t-tests were used to compute the significance values; the test did not return a significance value for non-professional designers' ratings for site 2.

Analysis of page-level data showed that participants preferred the modified versions of pages 57.4% of the time and preferred the original versions of pages 42.6% of the time; this difference was significant. Analysis of site-level data showed that participants rated the modified sites higher than the original sites, and this difference was significant. The site-level analysis also showed that the modified version of the site discussed in Chapter 8 was rated slightly higher than the original site; however, the difference was not significant. Participants' comments in both parts of the study provided support for many of the changes made to pages in the example site in Chapter 8 (and the other four) based on the Web interface profiles.

Modifications made to the pages and sites were very conservative for this first study. This is mainly because of the amount of effort required to manually make changes and because it was not clear if changes made based on the models actually improved quality; the latter was examined by the study itself. It is possible that less conservative changes would have resulted in larger differences in page preferences and site ratings. Future studies will be conducted to re-examine this question after recommendations and possibly modifications have been automated in some manner.

Chapter 10

Applying the Web Interface Profiles: Empirical Examination of Web Design Guidelines

10.1 Introduction

As discussed in Chapter 5, there are a huge number of Web design guidelines available in print and on the Web. Oftentimes these recommendations are vague, contradictory, and most have not been empirically validated. This chapter presents an analysis in which metric values in the profiles are compared against established guidelines for nine Web interface aspects listed below.

- Amount of text on a page
- Length and quality of link text
- Number and type of links on a page
- Use of non-animated and animated graphical ads
- Font styles and sizes
- Unique colors and color combinations
- Download speed
- Accessibility and HTML errors
- Consistency across pages

Many of the guidelines for these aspects relate to quantitative measures that play a major role in predicting page and site quality; these measures were discussed during profile development (Section 6.5.2) and in the assessment of the example site (Section 8.5). The profiles are used to derive thresholds (i.e., ranges of acceptable metric values) for measures that are relevant to the Web design aspects above. However, thresholds for individual measures are not intended to be used in isolation; as demonstrated by the example Web site assessment in Chapter 8, the models encapsulate relationships among one or more measures that may not be reflected by individual measures. The analysis below primarily uses the good page clusters (i.e., small-page, large-page,

and formatted-page), since they provide more context than the overall page quality model. In cases where relevant measures are not included in the cluster models, the analysis uses the overall page quality model instead. Similarly, the site-level analysis (consistency across pages) uses the overall site quality model; only site-level measures are used by this model and consequently reported in the analysis. In some cases the thresholds support current design guidelines, and in other cases they counter them.

The discussion in this chapter is not intended to imply that the profiles reflect causal links. For example, one limitation of the current models and tools is that they cannot improve on poor content. However, the study in the previous chapter provided preliminary evidence that they can provide insight on how to take good content that is poorly presented and improve its presentation, thus improving users' experience in accessing that content. And, because it is possible to empirically find commonalities among the presentation elements of the highly-rated sites, this provides strong evidence that the presentational aspects of highly-rated sites that differ from those of poorly-rated sites are in fact important for good design. The assessment in this chapter examines guidelines associated with some of these presentational aspects.

10.2 Page-Level Guidelines

The following sections summarize the comparison of thresholds derived from the good page cluster and overall page quality models to Web design guidelines. Eight page-level aspects are explored, including the amount of text on a page, the number and type of links, and color usage.

10.2.1 Amount of Text on a Page (Text Element)

The literature includes the following contradictory heuristics about the ideal amount of text for a Web page. Furthermore, there is no concrete guidance on how much text is enough or too much.

1. Users prefer pages with more content over breaking content into multiple pages [Landesman and Schroeder 2000].

2. Keep text short; use 50% less text than in print publications [Nielsen 2000].

3. Break text up into smaller units on multiple pages [Flanders and Willis 1998; Nielsen 2000].

The three good page clusters (discussed in Section 6.7) provide some insight about the ideal amount of text; however, the clusters do not provide insight on whether text is broken up into multiple pages on good sites. Table 10.1 depicts key text element and formatting measures with ranges based on the cluster centroids. The ranges suggest that pages with both a small and large amount of text are acceptable; however, the profile discussion revealed that text formatting needs to be proportional to the amount of text (see Section 6.10). For example, larger pages need to contain more headings than smaller pages to facilitate scrolling. Table 10.1 shows that headings (display word count), text clustering (text cluster count), as well as the number of columns where text starts on pages (text column count) varies for the three clusters. The ranges are all significantly different across clusters as determined by ANOVAs. Thus, the ranges in Table 10.1 appear to support all of the guidelines above, although it is not possible to assess where text is broken up into multiple pages.

One related question that is often posed, especially by novice designers, is whether or not home pages should be restricted to one scroll. The overall page quality models shows that good

Measure	Good Page Cluster		
	Small-Page	Large-Page	Formatted-Page
Word Count	72.8–371.1	710.4–1008.7	218.6–517.0
Display Word Count	1.1–18.7	20.8–38.4	14.7–32.2
Text Column Count	0.6–4.4	1.9–5.7	6.4–10.1
Text Cluster Count	0.0–2.3	1.9–4.2	2.6–4.9

Table 10.1: Word count and other text element and formatting ranges for good pages. The cluster models were used to derive ranges for all of the measures.

home pages typically require from 0.69 to 3.25 vertical scrolls, which suggests that it is not the case that home pages are restricted to one scroll. However, this may depend on the content category.

10.2.2 Length and Quality of Link Text (Text Element)

Nielsen [2000] suggests that Web designers use 2–4 words in text links; however, Sawyer and Schroeder [2000] suggest that Web designers use links with 7–12 "useful" words (i.e., words that provide hints about the content on a destination page). The average link words measure for all of the good pages suggests that text links on these pages contain from two to three words. Furthermore, the average good link words measure suggests that one to three of these text link words are not stop words or the word 'click,' which suggests that they are potentially useful. Ranges for each of the three good page clusters are very similar. Hence, the data suggests that link text on good pages is consistent with Nielsen's heuristic.

10.2.3 Number and Type of Links on a Page (Link Element)

Section 5.5 provided several guidelines on the number and type of links on a page as summarized below.

1. A large number of links impedes navigation [Spool *et al.* 1999].
2. Avoid using graphical text links; they are typically ignored [Sawyer and Schroeder 2000; Spool *et al.* 1999] or may impede navigation [Scanlon and Schroeder 2000c; Spool *et al.* 1999].
3. Use corresponding text links (for graphical links) [Flanders and Willis 1998; Sano 1996].
4. Avoid within-page links, since they may be confusing [Nielsen 2000; Sawyer and Schroeder 2000; Spool *et al.* 1999].
5. Use multiple links to the same content with appropriate scent in each area [Spool *et al.* 2000].
6. Use different forms for repeated links (e.g., text, graphical text, or image) [Sawyer and Schroeder 2000].
7. Redundant links may cause confusion [Kim and Yoo 2000].

Table 10.2 provides ranges for the link, text link, redundant link, and link graphic counts on pages in the three cluster models. The table also depicts ranges for link text cluster counts and shows that the number of link text clusters are somewhat proportional to the number of links on pages. For example, pages in the formatted-page cluster appear to have the most links as well

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Link Count	12.4–41.2	35.8–64.6	54.0–82.9	
Text Link Count	5.0–28.0	24.3–47.3	35.8–58.8	
Redundant Link Count	1.4–9.1	4.6–12.3	9.0–16.7	
Link Graphic Count	5.3–14.3	5.8–14.8	12.8–21.9	
Page Link Count				0.0–5.0
Link Text Cluster Count	0.0–1.4	0.5–1.9	1.7–3.1	

Table 10.2: Link element and text formatting ranges for good pages. The cluster models were used to derive ranges for all of the measures, except page link count; the overall page quality model was used to derive the range for this measure.

as the most link text clusters. The table shows that redundant links are used on good pages and graphical links are not avoided as suggested above; it is possible that the graphical links have text on them, which is not currently detected by the metrics tool. Given the ranges for redundant links along with the ranges for text and graphical links, it appears that links are repeated in both text and image formats. However, the measures do not reveal how often redundant links correspond to those that are also graphical links.

Table 10.2 shows that good pages also contain from zero to five within-page links. The means and standard deviations for each cluster suggest that within-page links are used infrequently on pages in the small-page and formatted-page clusters; however, they appear to be used more so on pages in the large-page cluster. Examining pages with many within-page links in this cluster revealed that most were reference pages (e.g., FAQs) with numerous links to allow users to return to the top portion of pages.

10.2.4 Use of Non-Animated and Animated Graphical Ads (Graphic Element)

The following guidance has been provided on the use of graphical ads and animation.

1. Ads affect the user experience; integrate ads with content [Klee and Schroeder 2000].
2. Usability dictates that ads should be eliminated [Nielsen 2000].
3. Ads increase credibility [Kim and Fogg 1999].
4. Minimize animated graphics [Flanders and Willis 1998].
5. Avoid using animation unless it is appropriate (e.g., showing transitions over time) [Nielsen 2000].
6. Animation is irritating to users; it impedes scanning [Spool *et al.* 1999].

Table 10.3 depicts ranges for animated graphical ads in the three clusters. Pages in each cluster are likely to contain one or more graphical ads, although pages in the formatted-page cluster typically contain the most graphical ads. Good pages tend to have zero or one animated graphical ad, which suggests that no more than one graphical ad is animated on good pages. Thus, it appears that animation is used sparingly.

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Graphic Ad Count	0.1–1.5	0.2–1.6	1.6–3.0	
Animated Graphic Ad Count				0.0–1.4

Table 10.3: Graphical ad ranges for good pages. The cluster models were used to derive ranges for the graphic ad count, while the overall page quality model was used to derive the range for the animated graphic ad count.

10.2.5 Font Styles and Sizes (Text and Page Formatting)

Guidelines on the use of font styles (serif or sans serif) and ideal font sizes for text include the following.

1. Use serif fonts for faster reading by older adults [Bernard *et al.* 2001].
2. Sans serif fonts have a slight advantage over serif fonts and are more preferred [Bernard and Mills 2000; Schriver 1997].
3. Use only a few sizes from one or two typeface families; use one serif and one sans serif font for contrast [Schriver 1997].
4. Use sans serif fonts for smaller text and serif fonts for larger text [Nielsen 2000].
5. Use 14 pt fonts for older adults [Bernard *et al.* 2001].
6. Use font sizes greater than 9 pt [Flanders and Willis 1998; Schriver 1997].
7. Use 10 to 11 pt (or higher) for body text and 14 pt (or higher) for display text; use larger point sizes for serif faces [Schriver 1997].

Table 10.4 summarizes ranges for other key font style and size measures on good pages. Based on the mean for the font style measure, sans serif is the predominant font style on good pages. However, the serif font count shows that these pages may use one serif font as well. In fact, the sans serif and serif font counts suggest that one serif font face and one sans serif font face is used on good pages. These pages also use at least three font combinations (font count) – font face, size, bolding, and italics – based on the font counts. The average font size used on pages in the three clusters is 9 pt or greater, which is somewhat consistent with the guidelines above. The pages also use a slightly smaller font size (minimum font size), typically for copyright or footer text.

Contrary to recommendations for using serif fonts for display text and sans serif fonts for body text, sans serif fonts appear to be used for both on good pages. There are large correlations between body and sans serif word counts as well as medium-strength correlations between display and sans serif word counts. There are no large correlations between display and serif word counts, suggesting that serif fonts may be used for something other than headings, such as form elements. The measures do not capture whether larger font sizes are used with serif fonts and vice versa for sans serif fonts.

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Font Style				Sans Serif
Font Count	3.6–5.8	5.7–7.9	5.8–8.0	
Sans Serif Font Count				0.4–1.2
Serif Font Count				0.1–1.3
Average Font Size	10.2–11.3	10.4–11.5	9.6–10.8	
Minimum Font Size	8.8–9.1	8.8–9.1	8.8–9.0	

Table 10.4: Font style and size ranges for good pages. The cluster models were used to derive ranges for half of the measures, and the overall page quality model was used to derive ranges for the other half. The font style measure reflects the predominant font face (serif, sans serif, or undetermined) used on a page.

10.2.6 Unique Colors and Color Combinations (Text, Link, and Page Formatting)

The literature offers the following guidance on the use of colors for text, links, and other Web page elements.

1. Minimize the number of text colors [Flanders and Willis 1998].
2. Use link and visited link colors that are similar to default browser colors (e.g., shades of blue, red, and purple) [Nielsen 2000].
3. Use default browser colors for links [Spool *et al.* 1999].
4. Use no more than 6 discriminable colors [Murch 1985].
5. Use browser-safe colors [Kaiser 1998].
6. Use color combinations determined to be good (i.e., high contrast) via user studies [Murch 1985].
7. Use high contrast between background and text [Flanders and Willis 1998; Nielsen 2000].

Table 10.5 summarizes ranges for various color measures. Good pages use from one to three colors for body text (body color count). Similarly, all three of the good page clusters use from one to three colors for display text (display color count); it appears that the number of display colors is proportional to the amount of text or formatting on pages. Although it is not clear whether different colors are used for body and display text on these pages, color counts for the three clusters suggest that this may be the case. Table 10.5 shows that the total number of colors used on good pages range from five to twelve with two or three of these colors being used for links and another one to three being used for body text; there is a gap between the total number of colors and the ranges for link and body text colors suggesting that the remaining distinct colors are used for display text. It seems that good pages may use up to six different colors for text, which appears to contradict the first heuristic for minimizing the number of text colors.

Table 10.5 shows that good pages use from two to four different colors for links (link color count). Furthermore, not all of these colors are standard link colors (standard link color count); the measures do not assess whether colors similar to the standard link colors are used. These ranges suggest that good pages do not closely follow the guidance in the literature on link colors.

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Body Color Count				0.9–3.1
Display Color Count	0.7–1.6	1.3–2.2	1.7–2.6	
Link Color Count				2.3–4.3
Standard Link Color Count	0.6–1.8	0.6–1.7	0.2–1.4	
Color Count	5.4–8.1	6.5–9.2	9.7–12.5	
Browser-Safe Color Count				3.5–6.6
Good Text Color Combinations	1.3–3.5	2.2–4.5	4.6–6.9	
Neutral Text Color Combinations				0.4–2.4
Bad Text Color Combinations				0.0–1.1
Good Panel Color Combinations	0.0–0.6	0.0–0.7	0.6–1.3	
Neutral Panel Color Combinations				0.0–1.2
Bad Panel Color Combinations	0.0–1.2	0.3–1.6	1.5–2.8	

Table 10.5: Color and color combination ranges for good pages. The cluster models were used to derive ranges for half of the measures, and the overall page quality model was used to derive ranges for the other half.

Ranges for the total number of unique colors (color count) on pages in each cluster show that good pages do not adhere to the guidance of using no more than six discriminable colors. Furthermore, they do not strictly use browser-safe colors (browser-safe color count) as recommended by the literature. Good pages tend to use good text color combinations more so than neutral and bad text color combinations. However, they tend to use neutral and bad panel color combinations (i.e., thick lines or shaded areas) more so than good panel color combinations. Inspections of good pages showed that oftentimes the panel color combinations are the reverse of text color combinations. For example, the page background may be white and a blue navigation bar may be placed on top of this white background, then white text is used on top of the blue navigation bar background. Based on studies by Murch [1985], the white background with the blue navigation bar is considered a neutral panel color combination, while the blue background with white text is considered a good text color combination. This discrepancy suggests that panel color combinations may not be as important as the text color combinations on pages, since they simply represent overlapping color regions, such as a colored navigation bar placed on top of the page's background color.

10.2.7 Download Speed (Page Performance)

The time for a page to fully load is considered a critical issue for Web interfaces, and the following guidance has been offered with respect to optimizing download speed.

1. Download speed should be no more than 10 seconds [Nielsen 2000].
2. Home pages greater than 40K result in significant bailouts [Zona Research 1999].
3. Keep graphic bytes to less than 35K [Flanders and Willis 1998].

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Download Time	12.1–20.7	11.9–20.5	10.2–18.8	
HTML Bytes	4.8–14.3K	15.4–24.9K	20.6–30.1K	
Graphic Bytes				4.4–59.0K
Script Bytes	248.0–1.7K	330.0–1.8K	1.1–2.5K	

Table 10.6: Download speed and byte ranges for good pages. The cluster models were used to derive ranges for all of the measures, except graphic bytes; the overall page quality model was used to derive the range for this measure.

4. Keep the Web page and all elements under 34K [Nielsen 2000].

Table 10.6 suggests that the estimated time to download good pages is rarely under ten seconds as suggested in the literature. However, the ranges are only a rough approximation of download speed and do not take into consideration caching of elements for subsequent use in the site. The download speed for good home pages (8.0–27.5 seconds) is most relevant (assuming users enter sites through home pages) and shows that indeed the estimated download speed is rarely under ten seconds; these estimates are based on a 41.2K connection speed[1]. The estimated download speed on average home pages ranges from 8.7 to 30.1 seconds, while the estimated download speed on poor home pages ranges from 7.2 to 27.4 seconds; this difference was not significant.

The literature also suggests that home pages greater than 40K result in greater bailout [Zona Research 1999]. HTML bytes on good home pages are from 11.8K to 33.3K, which is consistent with the guidance in the literature. Table 10.6 shows that bytes on all pages are below this recommended threshold. However, ranges for graphic bytes, HTML bytes, and script bytes in the table suggest that pages do not adhere to the guidance of keeping graphic bytes and the total bytes for all page elements below 34K. Script bytes are not reported since they are negligible for this data set.

10.2.8 Accessibility and HTML Errors (Page Performance)

Analysis in Section 6.6.2 revealed that Bobby and Weblint errors are more prevalent in good pages than average and poor pages despite the guidance that Web designers adhere to accessibility principles [Clark and Dardailler 1999; Cooper 1999; Nielsen 2000; Web Accessibility Initiative 1999] and avoid making HTML errors [Bowers 1996; Kim and Fogg 1999; Fogg *et al.* 2000]. Table 10.7 shows that pages in all clusters are typically not Bobby approved and contain several accessibility errors. Both the accessibility and Weblint errors appear to be proportional to the amount of formatting on pages; it was previously reported in Section 6.6.2 that these errors are correlated with the use of tables and interactive elements. The presence of accessibility and HTML coding errors on good pages is possibility attributable to the fact that HTML inherently does not afford designing accessible and error-free pages. Another possibility is that the tools are out of date.

[1]Section 5.12.4 showed that it is rarely possible to achieve a 56.6K connection speed with the 56.6K modem, possibly due to the technical limitations of this analog modem. For 50 connection sessions to three different Internet service providers at various times of the day, the average and median connection speed was found to be 41.2K. Hence, this connection speed is used by the download speed model.

Measure	Good Page Cluster			Overall Page
	Small-Page	Large-Page	Formatted-Page	
Bobby Approved	No	No	No	
Bobby Priority 1 Errors	0.6–1.6	0.8–1.8	1.5–2.5	
Bobby Priority 2 Errors	3.1–4.3	3.2–4.4	4.1–5.3	
Bobby Priority 3 Errors				1.8–2.2
Weblint Errors	0.0–36.2	16.4–53.0	55.8–92.4	

Table 10.7: Bobby and Weblint ranges for good pages. The cluster models were used to derive ranges for all of the measures, except Bobby priority 3 errors; the overall page quality model was used to derive the range for this measure. The Bobby approved measure reflects whether a page was Bobby approved (yes or no).

10.3 Site-Level Guidelines

The following section summarizes the comparison of thresholds derived from the overall site quality model to Web design guidelines on the consistency of pages throughout a site.

10.3.1 Consistency Across Pages (Site Architecture)

The consistency of page layout and page titles in the site has been discussed extensively in the literature, and the following guidance is provided.

1. Consistent layout of graphical interfaces result in a 10–25% speedup in performance [Mahajan and Shneiderman 1997].
2. Use consistent navigational elements [Flanders and Willis 1998; Fleming 1998].
3. Use several layouts (e.g., one for each page type) for variation within the site [Sano 1996].
4. Consistent elements become invisible [Sawyer *et al.* 2000].
5. Use different page titles for each page [Nielsen 2000].

Table 10.8 summarizes several consistency aspects of good sites. Recall that the variation measures are the median coefficients of variation ($100 * \frac{\sigma}{\bar{x}}$, where σ is the standard deviation, and $\bar{x}$ is the mean) computed across the measures within each category; larger variation suggests less consistency and vice versa. The page formatting variation suggests that page layouts are very consistent (no more than 22% variation) across pages on good sites; this is consistent with the first heuristic, but contradicts the fourth heuristic. The link element and formatting variation measures suggest that navigational elements are also fairly consistent across pages on good sites; this is consistent with the second heuristic. Finally, the page title variation suggests that page titles vary considerably on good sites, which supports the last heuristic above.

10.4 Summary

This chapter demonstrated the ability to apply the profiles of highly-rated interfaces for deriving guidelines based on empirical data and for validating existing guidelines. In many cases, such as the number and type of links, graphical ads, use of animation, color usage, download speed,

Measure	Overall Site
Page Formatting Variation	0.0–22.2%
Link Element Variation	35.2–101.2%
Link Formatting Variation	0.0–43.2%
Page Title Variation	19.5–179.5%

Table 10.8: Consistency ranges for good sites. The overall site quality model was used to derive these ranges.

and accessibility, the derived guidelines contradict heuristics from the literature. However, in some cases, such as the length of link text and font styles, the derived guidelines support heuristics from the literature. This approach can be used to develop empirically-derived guidelines for other Web interface aspects discussed in Chapter 5.

Chapter 11

Conclusions and Future Work

This dissertation makes several key contributions to the advancement of automated Web interface evaluation, including the following.

- It summarizes the current state of automated evaluation methods for graphical and Web interfaces and proposes ways to expand existing support.
- It describes how methods used for evaluating the performance of computer systems can be applied towards the problem of automated interface evaluation.
- It presents a methodology and tools for automated evaluation of Web interfaces that is a synthesis of approaches used in the usability and performance evaluation domains and based on empirical data.
- It describes an extensive set of quantitative Web interface measures for assessing many aspects discussed in the Web design literature.
- It describes the development of highly-accurate statistical models for assessing Web interface quality.
- It documents the efficacy of the statistical models for improving Web interface quality and for validating established Web design guidelines.

This dissertation presents the first research in which a large collection of expert-rated Web sites was analyzed to develop statistical models to support automated evaluation of new sites. This method represents a synthesis of approaches employed in the usability evaluation and performance evaluation domains. It entails computing an extensive set of 157 highly-accurate, quantitative page-level and site-level measures for sites that have been rated by Internet experts. These measures in conjunction with the expert ratings are used to derive statistical models of highly-rated Web interfaces. As is done with guideline review methods in the usability evaluation domain, the models are then used in the automated analysis of Web pages and sites. However, unlike other guideline review methods, the guidelines in this case are in essence derived from empirical data.

This dissertation shows that highly accurate models can be developed to assess Web page and site quality while taking into consideration the context in which pages and sites are designed. For example, models were developed to assess whether a page is a good home page or whether a page is consistent with pages on good health sites; site-level models were developed to enable similar assessments. A usability study suggests that the expert ratings used to derive the models are somewhat consistent with usability ratings; however, concrete conclusions cannot be drawn

from the study due to the time difference between expert and usability ratings. It was also shown that the models could be used to guide the modification of an example Web site, and another study showed that participants rated the modified version of this site slightly higher than the original version, although the difference was not significant. However, the study showed that for a larger set of sites, participants preferred pages modified based on the Web interface profiles over the original versions, and participants rated modified sites higher than the original sites; the differences were significant in both cases. Thus, this dissertation shows the first steps towards using the models to provide concrete Web design guidance.

Even though it was possible to find correlations between values for measures and expert ratings, no claim is being made about the profiles representing causal links. It is possible that the highly-rated sites are highly rated for reasons other than what is assessed with the measures, such as the quality of the content of the site. The current models and tools cannot improve on poor content. However, the study of modified sites provided preliminary evidence that they can provide insight on how to take good content that is poorly presented and improve its presentation, thus improving users' experience in accessing that content. And, because it is possible to empirically find commonalities among the presentation elements of the highly-rated sites, this provides strong evidence that the presentational aspects of highly-rated sites that differ from those of poorly-rated sites are in fact important for good design.

This dissertation represents an important first step towards enabling non-profes- sional designers to iteratively improve the quality of their Web designs. The methodology and tools are in their infancy and only provide support for refining an implemented site; thus, there are many ways in which they can be improved. Ideally, predictive models would be developed based on usability test results instead of expert ratings; however, a large effort needs to be launched within the HCI community to secure a large sample of usability tested sites. Developing and deploying models is very time and resource intensive; thus, automating some aspects of these activities would be extremely helpful. Even after developing and deploying models, considerable work needs to be done to fully understand and validate design principles gleaned from them.

The set of quantitative measures can also be expanded to measure other aspects discussed in this dissertation, such as the reuse of Web interface elements across pages in a site. A major limitation of the current set of measures is that they do not assess content quality. Future work will explore using text analysis techniques to possibly derive other measures of content quality. Another limitation of the measures is that they do not gauge accessibility for the disabled; it was shown that the good Web pages tended not to be accessible as determined by the Bobby tool. Future work will examine other ways to measure accessibility, for instance computing the nesting level of tables. (Although tables may help sighted users scan pages, they may impede blind users.)

Image processing could be used to improve the accuracy of existing measures, to enable the development of new ones, and to enable support for non-HTML pages and early design representations. Supporting early design representations also requires adjustments to be made to the profiles, such as ignoring certain measures during analysis.

All of the developed tools need to be reimplemented as part of a robust, open source browser, such as Mozilla or Opera; this will enable support for framesets, scripts, applets, and other objects as well as real-time analysis. Real-time analysis is crucial for developing an interactive evaluation tool to support iterative design and evaluation.

Other key components of the interactive evaluation tool include: recommending design improvements based on model predictions; applying recommendations so users can preview the changes; and showing comparable designs for exploration. Some of the model deviations are easy to correct, such as removing or changing text formatting, using good color combinations, and resizing images; it is possible to automatically modify the HTML to incorporate these types of changes.

Other changes, such as reducing vertical scrolling, adding text columns, improving readability, and adding links are not as straightforward. More work needs to be done to better understand the profiles before interactive evaluation can be supported. For example, factor analysis or multidimensional scaling techniques could be used to reduce the number of measures and to gain more insight about relationships among measures. This would enable recommendations to be based on combinations of measures versus individual measures.

A natural extension of this work is to enable profiles to be developed to capture effective design practices in other cultures. This would require support for non-English languages, mainly using language-appropriate dictionaries and text analysis algorithms. This may also require changes to the assessment of good color usage, since colors have specific meanings in some cultures.

Another extension of this work would be to develop a quality summary for Web pages and sites. This summary would represent a quick overview or characterization (possibly in graphical format) that could be used by search engines for ordering search results or possibly in visualizations of clickstreams through sites. The summary for search engine ordering may consist of a single rating, while the summary for the latter case may consist of a rating in addition to other details, such as the predicted page type, the closest good page cluster, and download speed.

Another application of the corpus of Web pages and sites is to enable designers to explore the collection to inform design. Ideally, characteristics of Web pages and sites could be represented in a way that facilitates easily identifying pages and sites that satisfy some design criteria, such as effective navigation schemes within health sites or good page layouts, color palettes, and site maps. Task-based search techniques that exploit metadata [Elliott 2001; English *et al.* 2001; Hearst 2000] should be helpful; metadata for Web interfaces could consist of the quantitative measures developed in this dissertation as well as others that describe for instance the size of the site, the type of site, page size, a page's functional type, elements on a page (e.g., navigation bars), as well as site ratings.

Another extension of this work is to use the profiles to derive parameters for the Web interface simulator proposed in this dissertation. Monte Carlo simulation was suggested as a way to automate Web site evaluation by mimicking users' information-seeking behavior and estimating navigation time, errors, etc. Similar to the guideline derivation, profiles could be used to derive simulation model parameters, such as thinking and reading times for pages. The cluster models could be used to derive timing estimates for these activities based on the average number of links, words, and other measures that reflect page complexity. User studies would be conducted to validate the baselines before incorporating them into the simulator.

This dissertation lays the foundation for a new approach to automated evaluation of interfaces wherein the interfaces themselves are analyzed as data. Continuous, ongoing analysis of interfaces will enable the models embedded in the approach to evolve and constantly reflect the current state of effective design. This approach is not intended to be used as a substitute for user input. Automated methods do not capture important qualitative and subjective information that can only be unveiled via usability testing and other inquiry methods. Furthermore, it is not the case that the issues identified by automated tools are true usability issues. Several studies, such as the one conducted by Bailey *et al.* [1992], have contrasted expert reviews and usability testing and found little overlap in findings between the two methods. Nonetheless, this approach should be a useful complement to non-automated evaluation techniques.

Bibliography

Abelow, David. 1993. Automating feedback on software product use. *CASE Trends* December.15–17.

Addy & Associates, 2000. Dr. Watson version 4.0. Available at http://watson.addy.com/.

Ahlberg, Christopher, and Ben Shneiderman. 1994. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the Conference on Human Factors in Computing Systems*, 313–317, Boston, MA. New York, NY: ACM Press.

Al-Qaimari, Ghassan, and Darren McRostie. 1999. KALDI: A computer-aided usability engineering tool for supporting testing and analysis of human-computer interaction. In *Proceedings of the 3rd International Conference on Computer-Aided Design of User Interfaces*, ed. by J. Vanderdonckt and A. Puerta, Louvain-la-Neuve, Belgium. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Ambühler, Reto, and Jakob Lindenmeyer. 1999. Measuring accessibility. In *Proceedings of the Eighth International World Wide Web Conference*, 48–49, Toronto, Canada. Foretec Seminars, Inc. Available at http://www.weboffice.ethz.ch/www8.

Anderson, J. 1993. *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, John R. 1990. *The Adaptive Character of Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Bacheldor, Beth. 1999. Push for performance. *Information Week* September 20.18–20.

Backus, J.W., F.L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A.J. Perlis, H. Rutishauer, K. Samelson, B. Vauquois, J.H. Wegstein, A. van Wijngaarden, and M. Woodger, 1964. Revised report on the algorithmic language algol 60. A/S regnecentralen, Copenhagen.

Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. New York, NY: ACM Press.

Bailey, Robert W., Robert W. Allan, and P. Raiello. 1992. Usability testing vs. heuristic evaluation: A head-to-head comparison. In *Proceedings of the Human Factors Society 36th Annual Meeting*, volume 1 of *COMPUTER SYSTEMS: Usability and Rapid Prototyping*, 409–413.

Balbo, Sandrine. 1995. Automatic evaluation of user interface usability: Dream or reality. In *Proceedings of the Queensland Computer-Human Interaction Symposium*, ed. by Sandrine Balbo, Queensland, Australia. Bond University.

——. 1996. ÉMA: Automatic analysis mechanism for the ergonomic evaluation of user interfaces. Technical Report 96/44, CSIRO Division of Information Technology. Available at http://www.cmis.csiro.au/sandrine.balbo/Ema/ema_tr/ema-tr.doc.

Barnard, Philip J. 1987. Cognitive resources and the learning of human-computer dialogs. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, ed. by John M. Carroll, 112–158. Cambridge, MA: MIT Press.

——, and J. D. Teasdale. 1991. Interacting cognitive subsystems: A systemic approach to cognitive-affective interaction and change. *Cognition and Emotion* 5.1–39.

Bash, Vinny La. 1997. 56.6k bps modems - not so fast! *Sarasota PC Monitor* July. Available at http://www.spcug.org/reviews/vl9707.htm.

Bauer, Christian, and Arno Scharl. 2000. Quantitative evaluation of web site content and structure. *Internet Research: Electronic Networking Applications and Policy* 10.31–43. Available at http://www.emerald-library.com.

Baumeister, Lynn K., Bonnie E. John, and Michael D. Byrne. 2000. A comparison of tools for building GOMS models. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 502–509, The Hague, The Netherlands. New York, NY: ACM Press.

Beard, David V., Dana K. Smith, and Kevin M. Denelsbeck. 1996. Quick and dirty GOMS: A case study of computed tomography interpretation. *Human-Computer Interaction* 11.157–180.

Bellotti, Victoria. 1988. Implications of current design practice for the use of HCI techniques. In *Proceedings of the HCI Conference on People and Computers IV*, 13–34, Manchester, United Kingdom. Amsterdam, The Netherlands: Elsevier Science Publishers.

Bernard, Michael, Chia Hui Liao, and Melissa Mills. 2001. The effects of font type and size on the legibility and reading time of online text by older adults. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 175–176, Seattle, WA.

——, and Melissa Mills. 2000. So, what size and type of font should I use on my website? *Usability News* Summer. Available at http://wsupsy.psy.twsu.edu/surl/usabilitynews/2S/font.htm.

Berners-Lee, Tim, 1995. Hypertext markup language - 2.0. http://www.ics.uci.edu/pub/ietf/html/rfc1866.txt.

Bevan, Nigel, and Miles Macleod. 1994. Usability measurement in context. *Behaviour and Information Technology* 13.132–145.

Bodart, F., A. M. Hennebert, J. M. Leheureux, and J. Vanderdonckt. 1994. Towards a dynamic strategy for computer-aided visual placement. In *Proceedings of the International Conference on Advanced Visual Interfaces*, ed. by T. Catarci, M. Costabile, M. Levialdi, and G. Santucci, 78–87, Bari, Italy. New York, NY: ACM Press.

Borges, Jose A., Israel Morales, and Nestor J. Rodriguez. 1996. Guidelines for designing usable world wide web pages. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 277–278, Vancouver, Canada. New York, NY: ACM Press.

Bourgin, David, 1994. Color space FAQ. http://www.neuro.sfc.keio.ac.jp/~aly/polygon/info/color-space-faq.html.

Bowers, Neil. 1996. Weblint: quality assurance for the World Wide Web. In *Proceedings of the Fifth International World Wide Web Conference*, Paris, France. Amsterdam, The Netherlands: Elsevier Science Publishers. Available at http://www5conf.inria.fr/fich_html/papers/P34/Overview.html.

Boyarski, Dan, Christine Neuwirth, Jodi Forlizzi, and Susan Harkness Regli. 1998. A study of fonts designed for screen display. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 87–94. New York, NY: ACM Press.

Brajnik, Giorgio. 2000. Automatic web usability evaluation: Where is the limit? In *Proceedings of the 6th Conference on Human Factors & the Web*, Austin, TX. Available at http://www.tri.sbc.com/hfweb/brajnik/hfweb-brajnik.html.

Breiman, Leo, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Belmont, California: Wadsworth Publishing Company.

Byrne, Michael D., Bonnie E. John, Neil S. Wehrle, and David C. Crow. 1999. The tangled web we wove: A taxonomy of WWW use. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 544–551, Pittsburg, PA. New York, NY: ACM Press.

——, Scott D. Wood, Piyawadee "Noi" Sukaviriya, James D. Foley, and David Kieras. 1994. Automating interface evaluation. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 232–237. New York, NY: ACM Press.

Card, Stuart K., Thomas P. Moran, and Allen Newell. 1983. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Centerline, 1999. QC/Replay. Available at http://www.centerline.com/productline/qcreplay/qcreplay.html.

Chak, Andrew. 2000. Usability tools: A useful start. *Web Techniques* August.18–20.

Cherry, L. L., and W. Vesterman. 1981. Writing tools: The STYLE and DICTION programs. Computer Science Technical Report 91, Bell Laboratories, Murray Hill, N.J. Republished as part of the 4.4BSD User's Supplementary Documents by O'Reilly.

Chi, Ed H., Peter Pirolli, Kim Chen, and James Pitkow. 2001. Using information scent to model user information needs and actions on the web. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 490–497, Seattle, WA. New York, NY: ACM Press.

——, Peter Pirolli, and James Pitkow. 2000. The scent of a site: A system for analyzing and predicting information scent, usage, and usability of a web site. In *Proceedings of the Conference on Human Factors in Computing Systems*, 161–168, The Hague, The Netherlands. New York, NY: ACM Press.

Clark, David, and Daniel Dardailler. 1999. Accessibility on the web: Evaluation & repair tools to make it possible. In *Proceedings of the CSUN Technology and Persons with Disabilities Conference*, Los Angeles, CA. Available at http://www.dinf.org/csun_99/session0030.html.

COLTHEART, M. 2001. The MRC psycholinguistic database. *Quarterly Journal of Experimental Psychology* 33.497–505.

COMBER, TIM. 1995. Building usable web pages: An HCI perspective. In *Proceedings of the First Australian World Wide Web Conference*, ed. by Roger Debreceny and Allan Ellis, 119–124, Ballina, Australia. Ballina, Australia: Norsearch. Available at http://www.scu.edu.au/sponsored/ausweb/ausweb95/papers/hypertext/comber/.

COOPER, MICHAEL. 1999. Universal design of a web site. In *Proceedings of the CSUN Technology and Persons with Disabilities Conference*, Los Angeles, CA. Available at http://www.dinf.org/csun_99/session0030.html.

COUTAZ, JOELLE. 1995. Evaluation techniques: Exploring the intersection of HCI and software engineering. In *Software Engineering and Human-Computer Interaction*, ed. by R. N. Taylor and J. Coutaz, Lecture Notes in Computer Science, 35–48. Heidelberg, Germany: Springer-Verlag.

CREATIVE GOOD, 1999. White paper one: Building a great customer experience to develop brand, increase loyalty and grow revenues. Available at http://www.creativegood.com/creativegood-whitepaper.pdf.

CUGINI, JOHN, and JEAN SCHOLTZ. 1999. VISVIP: 3D visualization of paths through web sites. In *Proceedings of the International Workshop on Web-Based Information Visualization*, 259–263, Florence, Italy. Institute of Electrical and Electronics Engineers.

DE HAAN, G., G. C. VAN DER VEER, and J. C. VAN VLIET. 1992. Formal modelling techniques in human-computer interaction. In *Cognitive Ergonomics: Contributions from Experimental Psychology*, ed. by Gerrit C. van der Veer, Sebastiano Bagnara, and Gerard A. M. Kempen, Theoretical Issues, 27–67. Amsterdam, The Netherlands: Elsevier Science Publishers.

DE SOUZA, FLAVIO, and NIGEL BEVAN. 1990. The use of guidelines in menu interface design: Evaluation of a draft standard. In *Proceedings of the Third IFIP TC13 Conference on Human-Computer Interaction*, ed. by G. Cockton, D. Diaper, and B. Shackel, 435–440, Cambridge, UK. Amsterdam, The Netherlands: Elsevier Science Publishers.

DESURVIRE, HEATHER W. 1994. Faster, cheaper!! are usability inspection methods as effective as empirical testing? In *Usability Inspection Methods*, ed. by Jakob Nielsen and Robert L. Mack. New York, NY: Wiley.

DETWEILER, MARK C., and RICHARD C. OMANSON, 1996. Ameritech web page user interface standards and design guidelines. Ameritech Corporation, Chicago, IL. Available at http://www.ameritech.com/corporate/testtown/library/standard/web_guidelines/index.html.

DIX, ALAN, JANET FINLAY, GREGORY ABOWD, and RUSSELL BEALE. 1998. *Human-Computer Interaction*. Upper Saddle River, NJ: Prentice Hall, second edition.

DOWD, KEVIN. 1993. *High Performance Computing*. Newton, MA: O'Reilly & Associates, Inc.

DREAMINK, 2000. Demographics and statistics for web designers – DreamInk web design tutorial. Available at http://dreamink.com/design5.shtml.

Drott, M. Carl. 1998. Using web server logs to improve site design. In *Proceedings of the 16th International Conference on Systems Documentation*, 43–50, Quebec, Canada. New York, NY: ACM Press.

Easton, Valerie J., and John H. McColl, 1997. Statistics glossary v1.1. Available at http://www.stats.gla.ac.uk/steps/glossary/index.html.

Elliott, Ame. 2001. Flamenco image browser: Using metadata to improve image search during architectural design. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 69–70, Seattle, WA.

English, Jennifer, Marti Hearst, Rashmi Sinha, Kirsten Swearingon, and Ping Yee, 2001. Examining the usability of web site search. Submitted for publication.

Etgen, Michael, and Judy Cantor. 1999. What does getting WET (Web Event-logging Tool) mean for web usability. In *Proceedings of the 5th Conference on Human Factors & the Web*, Gaithersburg, Maryland. Available at http://www.nist.gov/itl/div894/vvrg/hfweb/proceedings/etgen-cantor/index.html.

Faraday, Pete, and Alistair Sutcliffe. 1998. Providing advice for multimedia designers. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 124–131, Los Angeles, CA. New York, NY: ACM Press.

Faraday, Peter. 2000. Visually critiquing web pages. In *Proceedings of the 6th Conference on Human Factors & the Web*, Austin, TX. Available at http://www.tri.sbc.com/hfweb/faraday/faraday.htm.

Farenc, Christelle, Véronique Liberati, and Marie-France Barthet. 1999. Automatic ergonomic evaluation: What are the limits. In *Proceedings of the 3rd International Conference on Computer-Aided Design of User Interfaces*, Louvain-la-Neuve, Belgium. Dordrecht, The Netherlands: Kluwer Academic Publishers.

——, and Philippe Palanque. 1999. A generic framework based on ergonomics rules for computer aided design of user interface. In *Proceedings of the 3rd International Conference on Computer-Aided Design of User Interfaces*, ed. by J. Vanderdonckt and A. Puerta, Louvain-la-Neuve, Belgium. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Flanders, Vincent, and Michael Willis. 1998. *Web Pages That Suck: Learn Good Design by Looking at Bad Design*. San Francisco, CA: SYBEX.

Fleming, Jennifer. 1998. *Web Navigation: Designing the User Experience*. Sebastopol, CA: O'Reilly & Associates.

Fogg, B. J., Jonathan Marshall, Othman Laraki, Alex Osipovich, Chris Varma, Nicholas Fang, Jyoti Paul, Akshay Rangnekar, John Shon, Preeti Swani, and Marissa Treinen. 2001. What makes web sites credible?: a report on a large quantitative study. In *Proceedings of the ACM CHI'01 Conference on Human Factors in Computing Systems*, volume 1, 61–68, Seattle, WA.

Fogg, B.J., Jonathan Marshall, Alex Osipovich, Chris Varma, Othman Laraki, Nicholas Fang, Jyoti Paul, Akshay Rangnekar, John Shon, Preeti Swani, and

Marissa Treinen. 2000. Elements that affect web credibility: Early results from a self-report study. In *Proceedings of the Conference on Human Factors in Computing Systems*, number Extended Abstracts, 287–288, The Hague, The Netherlands. New York, NY: ACM Press. Available at http://www.webcredibility.org/WebCredEarlyResults.ppt.

Foley, James D., Andreis van Dam, Steven Feiner, and J. Hughes. 1990. *Computer Graphics: Principles and Practice*. Reading, MA: Addison-Wesley, 2nd edition.

Forrester Research, 1999. Why most web sites fail. Available at http://www.forrester.com/Research/ReportExcerpt/0,1082,1285,00.html.

Fuller, Rodney, and Johannes J. de Graaff. 1996. Measuring user motivation from server log files. In *Proceedings of the 2nd Conference on Human Factors & the Web*, Redmond, WA. Available at http://www.microsoft.com/usability/webconf/fuller/fuller.htm.

Furnas, George W. 1997. Effective view navigation. In *Proceedings of Conference on Human Factors in Computing Systems*, volume 1, 367–374, Atlanta, GA. New York, NY: ACM Press.

Glenn, F. A., S. M. Schwartz, and L. V. Ross, 1992. Development of a human operator simulator version v (HOS-V): Design and implementation. U.S. Army Research Institute for the Behavioral and Social Sciences, PERI-POX, Alexandria, VA.

GNU Software Foundation, 1999. Wget. Available at http://www.gnu.org/software/wget/wget.html.

Gunning, Robert. 1973. *The Technique of Clear Writing*. New York, NY: McGraw-Hill Book Company.

Guzdial, Mark, Paulos Santos, Albert Badre, Scott Hudson, and Mark Gray. 1994. Analyzing and visualizing log files: A computational science of usability. GVU Center TR GIT-GVU-94-8, Georgia Institute of Technology. Available at http://www.cc.gatech.edu/gvu/reports/1994/abstracts/94-08.html.

Hammontree, Monty L., Jeffrey J. Hendrickson, and Billy W. Hensley. 1992. Integrated data capture and analysis tools for research and testing on graphical user interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, 431–432, Monterey, CA. New York, NY: ACM Press.

Harper, B., and K. Norman. 1993. Improving user satisfaction: The questionnaire for user satisfaction interaction version 5.5. In *Proceedings of the 1st Annual Mid-Atlantic Human Factors Conference*, 224–228, Virginia Beach, VA.

Hartson, H. Rex, José C. Castillo, John Kelsa, and Wayne C. Neale. 1996. Remote evaluation: The network as an extension of the usability laboratory. In *Proceedings of the Conference on Human Factors in Computing Systems*, ed. by Michael J. Tauber, Victoria Bellotti, Robin Jeffries, Jock D. Mackinlay, and Jakob Nielsen, 228–235, Vancouver, Canada. New York, NY: ACM Press.

Healey, Christopher G. 1996. A perceptual colour segmentation algorithm. Technical Report TR-96-09, University of British Columbia, Department of Computer Science.

Hearst, Marti A. 2000. Next generation web search: Setting our sites. *IEEE Data Engineering Bulletin* 23.

Helfrich, Brian, and James A. Landay, 1999. QUIP: quantitative user interface profiling. Unpublished manuscript. Available at http://home.earthlink.net/~bhelfrich/quip/index.html.

Hochheiser, Harry, and Ben Shneiderman. 2001. Using interactive visualizations of WWW log data to characterize access patterns and inform site design. *Journal of the American Society for Information Science and Technology* 52.331–343.

Hoffman, Donna L., Thomas P. Novak, and Patrali Chatterjee. 1995. Commercial scenarios for the web: Opportunities and challenges. *Journal of Computer-Mediated Communication* 3. Available at http://jcmc.huji.ac.il/vol1/issue3/hoffman.html.

Hom, James, 1998. The usability methods toolbox. Available at http://www.best.com/~jthom/usability/usable.htm.

Hong, Jason I., Jeffrey Heer, Sarah Waterson, and James A. Landay. 2001. Webquilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems* . To appear.

Hudson, Scott E., Bonnie E. John, Keith Knudsen, and Michael D. Byrne. 1999. A tool for creating predictive performance models from user interface demonstrations. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, 92–103, Asheville, NC. New York, NY: ACM Press.

Human Factors Engineering, 1999a. Ask usability advisor. Available at http://www.cs.umd.edu/~zzj/Advisor.html.

——, 1999b. Usability evaluation methods. Available at http://www.cs.umd.edu/~zzj/UsabilityHome.html.

International Standards Organization, 1999. Ergonomic requirements for office work with visual display terminals, part 11: Guidance on usability. Available from http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16883&ICS1=13&ICS2=180&ICS3=.

Internet Software Consortium, 2001. Internet domain survey. Available at http://www.isc.org/ds/.

Ivory, Melody Y. 2000. Web TANGO: Towards automated comparison of information-centric web site designs. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 329–330, The Hague, The Netherlands.

——, and Marti A. Hearst. 2001. State of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys* . To appear.

——, and ——. 2002. Statistical profiles of highly-rated web site interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, Minneapolis, MN. To appear.

——, Rashmi R. Sinha, and Marti A. Hearst. 2000. Preliminary findings on quantitative measures for distinguishing highly rated information-centric web pages. In *Proceedings of the 6th Conference on Human Factors & the Web*, Austin, TX. Available at http://www.tri.sbc.com/hfweb/ivory/paper.html.

——, ——, and ——. 2001. Empirically validated web page design metrics. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 53–60, Seattle, WA.

Jackson, Richard, Lindsay MacDonald, and Ken Freeman. 1994. *Computer Generated Color: A practical guide to presentation and display*. New York, NY: John Wiley & Sons.

Jain, R. 1991. *Human-Computer Interaction*. New York, NY: Wiley-Interscience.

Jasper, R., M. Brennan, K. Williamson, W. Currier, and D. Zimmerman. 1994. Test data generation and feasible path analysis. In *International Symposium on Software Testing and Analysis*, Seattle, WA.

Jeffries, Robin, James R. Miller, Cathleen Wharton, and Kathy M. Uyeda. 1991. User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of the Conference on Human Factors in Computing Systems*, 119–124, New Orleans, LA. New York, NY: ACM Press.

Jiang, J., E. Murphy, and L. Carter. 1993. Computer-human interaction models (CHIMES): Revision 3. Technical Report DSTL-94-008, National Aeronautics and Space Administration.

John, Bonnie E., and David E. Kieras. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction* 3.320–351.

Kaiser, Jean. 1998. Browser-safe colors. *Web Design* June 15. Available at http://webdesign.about.com/compute/webdesign/library/weekly/aa061598.htm.

Karlgren, Jussi, 2000. *Stylistic Experiments for Information Retrieval*. Department of Linguistics: Stockholm University dissertation. Available at http://www.sics.se/∼jussi/Artiklar/2000_PhD/.

Kasik, David J., and Harry G. George. 1996. Toward automatic generation of novice user test scripts. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 244–251, Vancouver, Canada. New York, NY: ACM Press.

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

Keppel, Geoffrey, and Sheldon Zedeck. 1989. *Data Analysis for Research Designs: Analysis-of-variance and Multiple Regression/Correlation Approaches*. A Series of Books in Psychology. New York, NY: W. H. Freeman.

Kieras, David, and Peter G. Polson. 1985. An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies* 22.365–394.

Kieras, David E., Scott D. Wood, Kasem Abotel, and Anthony Hornof. 1995. GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the 8th ACM Symposium on User Interface Software and Technology*, 91–100, Pittsburgh, PA. New York, NY: ACM Press.

——, Scott D. Wood, and David E. Meyer. 1997. Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction* 4.230–275.

Kim, Jinwoo, and Byunggon Yoo. 2000. Toward the optimal link structure of the cyber shopping mall. *International Journal of Human-Computer Studies* 52.531–551.

Kim, N., and B. J. Fogg, 1999. World wide web credibility: What effects do advertisements and typos have on the perceived credibility of web page information? Unpublished thesis, Stanford University.

Kim, Won Chul, and James D. Foley. 1993. Providing high-level control and expert assistance in the user interface presentation design. In *Proceedings of the Conference on Human Factors in Computing Systems*, ed. by Stacey Ashlund, Ken Mullet, Austin Henderson, Erik Hollnagel, and Ted White, 430–437, Amsterdam, The Netherlands. New York, NY: ACM Press.

Kirakowski, Jurek, and Nigel Claridge. 1998. Human centered measures of success in web site design. In *Proceedings of the 4th Conference on Human Factors & the Web*, Basking Ridge, NJ. Available at http://www.research.att.com/conf/hfweb/proceedings/kirakowski/index.html.

Klee, Matthew, and Will Schroeder. 2000. Report 2: How business goals affect site design. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

Laird, John E., and Paul Rosenbloom. 1996. The evolution of the Soar cognitive architecture. In *Mind Matters: A Tribute to Allen Newell*, ed. by David M. Steier and Tom M. Mitchell, 1–50. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Landesman, Lori, and Will Schroeder. 2000. Report 5: Organizing links. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

Larson, Kevin, and Mary Czerwinski. 1998. Web page design: Implications of memory, structure and scent for information retrieval. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 25–32, Los Angeles, CA. New York, NY: ACM Press.

Law, Averill M, and David W Kelton. 1991. *Simulation Modeling and Analysis*. Industrial Engineering and Management Science. McGraw-Hill International Edition.

Lecerof, Andreas, and Fabio Paternò. 1998. Automatic support for usability evaluation. *IEEE Transactions on Software Engineering* 24.863–888.

Lee, Kenton, 1997. Motif FAQ. Available at http://www-bioeng.ucsd.edu/~fvetter/misc/Motif-FAQ.txt.

Levine, Rick, 1996. Guide to web style. Sun Microsytems. Available at http://www.sun.com/styleguide/.

Lewis, Clayton, Peter G. Polson, Cathleen Wharton, and John Rieman. 1990. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, 235–242, Seattle, WA. New York, NY: ACM Press.

Lohse, Gerald L., and Peter Spiller. 1998. Quantifying the effect of user interface design features on cyberstore traffic and sales. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 211–218, Los Angeles, CA. New York, NY: ACM Press.

Lowgren, Jonas, and Tommy Nordqvist. 1992. Knowledge-based evaluation as design support for graphical user interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, 181–188, Monterey, CA. New York, NY: ACM Press.

Lynch, Gene, Susan Palmiter, and Chris Tilt. 1999. The max model: A standard web site user model. In *Proceedings of the 5th Conference on Human Factors & the Web*, Gaithersburg, Maryland. Available at http://www.nist.gov/itl/div894/vvrg/hfweb/proceedings/lynch/index.html.

Lynch, Patrick J., and Sarah Horton. 1999. *Web Style Guide: Basic Design Principles for Creating Web Sites*. Princeton, NJ: Yale University Press. Available at http://info.med.yale.edu/caim/manual/.

Macleod, Miles, Rosemary Bowden, Nigel Bevan, and Ian Curson. 1997. The MUSiC performance measurement method. *Behaviour and Information Technology* 16.279–293.

——, and Ralph Rengger. 1993. The development of DRUM: A software tool for video-assisted usability evaluation. In *Proceedings of the HCI Conference on People and Computers VIII*, 293–309, Loughborough, UK. Cambridge University Press, Cambridge, UK.

Mahajan, Rohit, and Ben Shneiderman. 1997. Visual and textual consistency checking tools for graphical user interfaces. *IEEE Transactions on Software Engineering* 23.722–735.

May, J., and P. J. Barnard. 1994. Supportive evaluation of interface design. In *Proceedings of the First Interdisciplinary Workshop on Cognitive Modeling and User Interface Design*, ed. by C. Stary, Vienna, Austria.

Mercury Interactive, 2000. Winrunner. Available at http://www-svca.mercuryinteractive.com/products/winrunner/.

Miller, Craig S., and Roger W. Remington. 2000. A computational model of web navigation: Exploring interactions between hierarchical depth and link ambiguity. In *Proceedings of the 6th Conference on Human Factors & the Web*, Austin, TX. Available at http://www.tri.sbc.com/hfweb/miller/article.html.

Molich, R., N. Bevan, S. Butler, I. Curson, E. Kindlund, J. Kirakowski, and D. Miller. 1998. Comparative evaluation of usability tests. In *Proceedings of the UPA Conference*, 189–200, Washington, DC. Usability Professionals' Association, Chicago, IL.

——, A. D. Thomsen, B. Karyukina, L. Schmidt, M. Ede, W. van Oel, and M. Arcuri. 1999. Comparative evaluation of usability tests. In *Proceedings of the Conference on Human Factors in Computing Systems*, 83–86, Pittsburg, PA. New York, NY: ACM Press.

Moran, Thomas P. 1981. The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies* 15.3–50.

—— 1983. Getting into a system: External-internal task mapping analysis. In *Proceedings of the Conference on Human Factors in Computing Systems*, 45–49, Boston, MA. New York, NY: ACM Press.

Murch, Gerald M. 1985. Colour graphics — blessing or ballyhoo? *Computer Graphics Forum* 4.127–135.

Myers, Brad A. 1992. Demonstrational interfaces: A step beyond direct manipulation. *IEEE Computer* 25.61–73.

Neal, Alan S., and Roger M. Simons. 1983. Playback: A method for evaluating the usability of software and its documentation. In *Proceedings of the Conference on Human Factors in Computing Systems*, 78–82, Boston, MA. New York, NY: ACM Press.

NetRaker, 2000. The NetRaker suite. Available at http://www.netraker.com/info/applications/index.asp.

Newman, Mark W., and James A. Landay. 2000. Sitemaps, storyboards, and specifications: A sketch of web site design practice. In *Proceedings of Designing Interactive Systems: DIS 2000*, Automatic Support in Design and Use, 263–274.

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

——, 1996. Top ten mistakes in web design. Available at http://www.useit.com/alertbox/9605.html.

——, 1997. Writing for the web. Available at http://www.useit.com/papers/webwriting.

——, 1998a. Cost of user testing a site. Available at http://www.useit.com/alertbox/980503.html.

——, 1998b. Using link titles to help users predict where they are going. Available at http://www.useit.com/alertbox/980111.html.

——. 1998c. Web usability: Why and how. *Users First!* September 14. Available at www.zdnet.com/devhead/stories/articles/0,4413,2137433,00.html.

——, 1999a. The top-10 new mistakes of web design. Available http://www.useit.com/alertbox/990530.html.

——. 1999b. User interface directions for the Web. *Communications of the ACM* 42.65–72.

——. 2000. *Designing Web Usability: The Practice of Simplicity*. Indianapolis, IN: New Riders Publishing.

——, and Robert L. Mack (eds.) 1994. *Usability Inspection Methods*. New York, NY: Wiley.

Olsen, Jr., Dan. R. 1992. *User Interface Management Systems: Models and Algorithms*. San Mateo, CA: Morgan Kaufman Publishers, Inc.

Olsen, Jr., Dan R., and Bradley W. Halversen. 1988. Interface usage measurements in a user interface management system. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, 102–108, Alberta, Canada. New York, NY: ACM Press.

Open Software Foundation. 1991. *OSF/Motif Style Guide*. Number Revision 1.1 (for OSF/Motif release 1.1). Englewood Cliffs, NJ: Prentice Hall.

Palanque, Philippe, Christelle Farenc, and Rémi Bastide. 1999. Embedding ergonomic rules as generic requirements in the development process of interactive software. In *Proceedings of IFIP TC13 Seventh International Conference on Human-Computer Interaction*, ed. by A. Sasse and C. Johnson, Edinburgh, Scotland. Amsterdam, The Netherlands: IOS Press.

Parush, Avraham, Ronen Nadir, and Avraham Shtub. 1998. Evaluating the layout of graphical user interface screens: Validation of a numerical, computerized model. *International Journal of Human Computer Interaction* 10.343–360.

Paternò, F., C. Mancini, and S. Meniconi. 1997. ConcurTaskTrees: Diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Sixth International Conference on Human-Computer Interaction*, 362–369, Sydney, Australia. Sydney, Australia: Chapman and Hall.

Paternò, Fabio, and Giulio Ballardin. 1999. Model-aided remote usability evaluation. In *Proceedings of the IFIP TC13 Seventh International Conference on Human-Computer Interaction*, ed. by A. Sasse and C. Johnson, 434–442, Edinburgh, Scotland. Amsterdam, The Netherlands: IOS Press.

Payne, Stephen J., and T. R. G. Green. 1986. Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction* 2.93–133.

Peck, Virginia A., and Bonnie E. John. 1992. Browser-soar: A computational model of a highly interactive task. In *Proceedings of the Conference on Human Factors in Computing Systems*, 165–172, Monterey, CA. New York, NY: ACM Press.

Petri, C. A. 1973. Concepts of net theory. In *Mathematical Foundations of Computer Science: Proceedings of the Symposium and Summer School*, 137–146, High Tatras, Czechoslovakia. Mathematical Institute of the Slovak Academy of Sciences.

Pew, R. W., and A. S. Mavor (eds.) 1998. *Modeling Human and Organizational Behavior: Application to Military Simulations*. Washington, DC: National Academy Press. Available at http://books.nap.edu/html/model.

Phelps, Thomas A., 1998. *Multivalent Documents: Anytime, Anywhere, Any Type, Every Way User-Improvable Digital Documents and Systems*. Computer Science Division: University of California, Berkeley dissertation.

——, 2001. The multivalent browser. Available at http://http.cs.berkeley.edu/ phelps/Multivalent/.

Pirolli, Peter. 1997. Computational models of information scent-following in a very large browsable text collection. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 1, 3–10, Atlanta, GA. New York, NY: ACM Press.

——, and Stuart Card. 1995. Information foraging in information access environments. In *Proceedings of the Conference on Human Factors in Computing Systems*, ed. by Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen, 51–58, New York, NY. ACM Press.

——, James Pitkow, and Ramana Rao. 1996. Silk from a sow's ear: Extracting usable structure from the web. In *Proceedings of the Conference on Human Factors in Computing Systems: Common Ground*, 118–125.

Polk, T. A., and P. S. Rosenbloom. 1994. Task-independent constraints on a unified theory of cognition. In *Handbook of Neuropsychology, Volume 9*, ed. by F. Boller and J. Grafman. Amsterdam, The Netherlands: Elsevier Science Publishers.

Porteous, M., J. Kirakowski, and M. Corbett. 1993. SUMI user handbook. Technical report, Human Factors Research Group, University College Cork, Ireland.

Ratner, Julie, Eric M. Grose, and Chris Forsythe. 1996. Characterization and assessment of HTML style guides. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 115–116, Vancouver, Canada. New York, NY: ACM Press.

Rauterberg, Matthias. 1995. From novice to expert decision behaviour: a qualitative modeling approach with petri nets. In *Symbiosis of Human and Artifact: Human and Social Aspects of Human-Computer Interaction*, ed. by Y. Anzai, K. Ogawa, and H. Mori, volume 20B of *Advances in Human Factors/Ergonomics*, 449–454. Amsterdam, The Netherlands: Elsevier Science Publishers.

——. 1996a. How to measure and to quantify usability of user interfaces. In *Advances in Applied Ergonomics*, ed. by A. Özok and G. Salvendy, 429–432. West Lafayette, IN: USA Publishing.

——. 1996b. A petri net based analyzing and modeling tool kit for logfiles in human-computer interaction. In *Proceedings of Cognitive Systems Engineering in Process Control*, 268–275, Kyoto, Japan. Kyoto University: Graduate School of Energy Science.

——, and Roger Aeppili. 1995. Learning in man-machine systems: the measurement of behavioural and cognitive complexity. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 4685–4690, Vancouver, BC. Institute of Electrical and Electronics Engineers.

Reisner, P. 1984. Formal grammar as a tool for analyzing ease of use: Some fundamental concepts. In *Human Factors in Computer Systems*, ed. by John C. Thomas and Michael L. Schneider, 53–78. Norwood, NJ: Ablex Publishing Corporation.

Reiterer, H. 1994. A user interface design assistant approach. In *Proceedings of the IFIP 13th World Computer Congress*, ed. by Klaus Brunnstein and Eckart Raubold, volume 2, 180–187, Hamburg, Germany. Amsterdam, The Netherlands: Elsevier Science Publishers.

Rengger, R., M. Maclead, R. Bowden, A. Drynan, and M. Blayney. 1993. MUSiC performance measurement handbook, v2. Technical report, National Physical Laboratory, Teddington, UK.

Rieman, John, Susan Davies, D. Charles Hair, Mary Esemplare, Peter Polson, and Clayton Lewis. 1991. An automated cognitive walkthrough. In *Proceedings of the Conference on Human Factors in Computing Systems*, 427–428, New Orleans, LA. New York, NY: ACM Press.

Roche, Xavier, 2001. Httrack website copier – offline browser. Available at http://www.httrack.com/.

Rosenfeld, Louis, and Peter Morville. 1998. *Information Architecture for the World Wide Web*. Sebastopol, CA: O'Reilly & Associates.

Rossi, Gustavo, Daniel Schwabe, and Fernando Lyardet. 1999. Improving web information systems with navigation patterns. In *Proceedings of the Eighth International World Wide Web Conference*, 48–49, Toronto, Canada. Foretec Seminars, Inc. Available at http://www8.org/w8-papers/5b-hypertext-medi/improving/improving.html.

Rowley, David E., and David G. Rhoades. 1992. The cognitive jogthrough: A fast-paced user interface evaluation procedure. In *Proceedings of the Conference on Human Factors in Computing Systems*, 389–395, Monterey, CA. New York, NY: ACM Press.

Sano, Darrell. 1996. *Designing Large-scale Web Sites: A Visual Design Methodology*. New York, NY: Wiley Computer Publishing, John Wiley & Sons, Inc.

Sauer, C. H., and K. M. Chandy. 1981. *Computer Systems Performance Modeling*. Englewood Cliffs, NJ: Prentice Hall.

Sawyer, Paul, Richard Danca, and Will Schroeder. 2000. Report 6: Myths of page layout. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

——, and Will Schroeder. 2000. Report 4: Links that give off scent. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

Scanlon, Tara, and Will Schroeder. 2000a. Report 1: What people do with web sites. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

——, and ——. 2000b. Report 10: On-site searching and scent. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

——, and ——. 2000c. Report 7: Designing graphics with a purpose. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

Scapin, Dominique, Corinne Leulier, Jean Vanderdonckt, Céline Mariage, Christian Bastien, Christelle Farenc, Philippe Palanque, and Rémi Bastide. 2000. A framework for organizing web usability guidelines. In *Proceedings of the 6th Conference on Human Factors & the Web*, Austin, TX. Available at http://www.tri.sbc.com/hfweb/scapin/Scapin.html.

Schaffer, Cullen. 1993. Selecting a classification method by cross-validation. *Machine Learning* 13.135–143.

Scholtz, Jean, and Sharon Laskowski. 1998. Developing usability tools and techniques for designing and testing web sites. In *Proceedings of the 4th Conference on Human Factors & the Web*, Basking Ridge, NJ. Available at http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html.

Schriver, Karen A. 1997. *Dynamics in Document Design*. New York, NY: Wiley Computer Publishing, John Wiley & Sons, Inc.

Schwartz, Matthew. 2000. Web site makeover. *Computerworld* January 31. Available at http://www.computerworld.com/home/print.nsf/all/000126e3e2.

Sears, Andrew. 1995. AIDE: A step toward metric-based interface development tools. In *Proceedings of the 8th ACM Symposium on User Interface Software and Technology*, 101–110, Pittsburg, PA. New York, NY: ACM Press.

Service Metrics, 1999. Service metrics solutions. Available at http://www.servicemetrics.com/solutions/solutionsmain.asp.

Shedroff, Nathan, 1999. Recipe for a successful web site. Available at http://www.nathan.com/thoughts/recipe.

——. 2001. *Experience Design 1.* Indianapolis, IN: New Riders Publishing.

Shneiderman, Ben. 1997. Designing information-abundant web sites: Issues and recommendations. *International Journal of Human-Computer Studies* 47.5–29.

——. 1998. *Designing the user interface: strategies for effective human-computer interaction.* Reading, MA: Addison-Wesley, third edition.

Sinha, Rashmi, Marti Hearst, and Melody Ivory. 2001. Content or graphics? an empirical analysis of criteria for award-winning websites. In *Proceedings of the 7th Conference on Human Factors & the Web*, Madison, WI.

Siochi, Antonio C., and Deborah Hix. 1991. A study of computer-supported user interface evaluation using maximal repeating pattern analysis. In *Proceedings of the Conference on Human Factors in Computing Systems*, 301–305, New Orleans, LA. New York, NY: ACM Press.

Smith, Sidney L. 1986. Standards versus guidelines for designing user interface software. *Behaviour and Information Technology* 5.47–61.

——, and Jane N. Mosier. 1986. Guidelines for designing user interface software. Technical Report ESD-TR-86-278, The MITRE Corporation, Bedford, MA 01730.

Spool, Jared, and Will Schroeder. 2001. Testing web sites: Five users is nowhere near enough. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 2, 285–286, Seattle, WA.

Spool, Jared M., Matthew Klee, and Will Schroeder. 2000. Report 3: Designing for scent. In *Designing Information-Rich Web Sites*. Bradford, MA: User Interface Engineering.

——, Tara Scanlon, Will Schroeder, Carolyn Snyder, and Terri DeAngelo. 1999. *Web Site Usability: A Designer's Guide.* San Francisco, CA: Morgan Kaufmann Publishers, Inc.

SPSS Inc. 1999. *SPSS Base 10.0 Applications Guide.* Chicago, IL: SPSS Inc.

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

Streveler, Dennis J., and Anthony I. Wasserman. 1984. Quantitative measures of the spatial properties of screen designs. In *Proceedings of the IFIP TC13 First International Conference on Human-Computer Interaction*, ed. by B. Shackel, 81–89, London, UK. Amsterdam, The Netherlands: North-Holland.

Sullivan, Terry. 1997. Reading reader reaction: A proposal for inferential analysis of web server log files. In *Proceedings of the 3rd Conference on Human Factors & the Web*, Boulder, CO. Available at http://www.research.att.com/conf/hfweb/conferences/denver3.zip.

Tauber, Michael J. 1990. ETAG: Extended task action grammar – A language for the description of the user's task language. In *Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction*, ed. by G. Cockton, D. Diaper, and B. Shackel, 163–168, Cambridge, UK. Amsterdam, The Netherlands: Elsevier Science Publishers.

The International Academy of Arts and Sciences, 2000. The webby awards 2000 judging criteria. Available at http://www.webbyawards.com/judging/criteria.html.

Theng, Yin Leng, and Gil Marsden. 1998. Authoring tools: Towards continuous usability testing of web documents. In *Proceedings of the 1st International Workshop on Hypermedia Development*, Pittsburg, PA. Available at http://www.eng.uts.edu.au/~dbl/HypDev/ht98w/YinLeng/HT98_YinLeng.html.

Thimbleby, Harold. 1997. Gentler: A tool for systematic web authoring. *International Journal of Human-Computer Studies* 47.139–168.

Tullis, T. S. 1983. The formatting of alphanumeric displays: A review and analysis. *Human Factors* 25.657–682.

Uehling, Dana L., and Karl Wolf. 1995. User action graphing effort (UsAGE). In *Proceedings of the Conference on Human Factors in Computing Systems*, Denver, CO. New York, NY: ACM Press.

Usable Net, 2000. LIFT online. Available at http://www.usablenet.com.

Vanderdonckt, J., and X. Gillo. 1994. Visual techniques for traditional and multimedia layouts. In *Proceedings of the International Conference on Advanced Visual Interfaces*, ed. by T. Catarci, M. Costabile, M. Levialdi, and G. Santucci, 95–104, Bari, Italy. New York, NY: ACM Press.

Web Accessibility Initiative, 1999. Web content accessibility guidelines 1.0. World Wide Web Consortium, Geneva, Switzerland. Available at http://www.w3.org/TR/WAI-WEBCONTENT/.

Web Criteria, 1999. Max, and the objective measurement of web sites. Available at http://www.webcriteria.com.

WebTrends Corporation, 2000. Webtrends live. http://www.webtrendslive.com/default.htm.

Weicker, Reinhold P. 1990. An overview of common benchmarks. *Computer* 23.65–75.

Whitefield, Andy, Frank Wilson, and John Dowell. 1991. A framework for human factors evaluation. *Behaviour and Information Technology* 10.65–79.

Williams, Kent E. 1993. Automating the cognitive task modeling process: An extension to GOMS for HCI. In *Proceedings of the Conference on Human Factors in Computing Systems*, volume 3, p. 182, Amsterdam, The Netherlands. New York, NY: ACM Press.

World Wide Web Consortium, 2000. HTML validation service. Available at http://validator.w3.org/.

Young, Richard M., T. R. G. Green, and Tony Simon. 1989. Programmable user models for predictive evaluation of interface designs. In *Proceedings of the Conference on Human Factors in Computing Systems*, 15–19, Austin, TX. New York, NY: ACM Press.

Zachary, W., J.-C. Le Mentec, and J. Ryder. 1996. Interface agents in complex systems. In *Human Interaction With Complex Systems: Conceptual Principles and Design Practice*, ed. by C. N. Ntuen and E. H. Park. Kluwer Academic Publishers: Dordrecht, The Netherlands.

Zaphiris, Panayiotis, and Lianaeli Mtei, 1997. Depth vs. breadth in the arrangement of Web links. Available at http://www.otal.umd.edu/SHORE/bs04.

Zettlemoyer, Luke S., Robert St. Amant, and Martin S. Dulberg. 1999. IBOTS: Agent control through the user interface. In *Proceedings of the International Conference on Intelligent User Interfaces*, 31–37, Redondo Beach, CA. New York, NY: ACM Press.

Zhang, Zhijun, Victor Basili, and Ben Shneiderman. 1998. An empirical study of perspective-based usability inspection. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting*, 1346–1350, Chicago, IL. Santa Monica, CA: Human Factors Ergonomics Society.

Zona Research, Inc., 1999. The economic impacts of unacceptable web site download speeds.

Appendix A

Automation Characteristics of UE Methods, Chapter 2

The following sections discuss automation characteristics for WIMP and Web interfaces separately. This information was aggregated and presented in Section 2.3.

A.1 Automation Characteristics of WIMP UE Methods

Tables A.1 and A.2 summarize automation characteristics for the 75 UE methods surveyed for WIMP interfaces. Table A.3 provides descriptions of all method types, and Table A.4 summarizes the number of non-automated and automated capture, analysis, and critique methods surveyed. Four software tools provide automation support for multiple method types: DRUM - performance measurement and log file analysis; AMME - log file analysis and petri net modeling; KALDI - performance measurement, log file analysis, and remote testing; and UsAGE - performance measurement and log file analysis.

Method Class / Method Type	**Automation Type**			
	None	**Capture**	**Analysis**	**Critique**
Testing				
Thinking-aloud Protocol	F (1)			
Question-asking Protocol	F (1)			
Shadowing Method	F (1)			
Coaching Method	F (1)			
Teaching Method	F (1)			
Co-discovery Learning	F (1)			
Performance Measurement	F (1)	F (4)		
Log File Analysis			IFM (10)*	
Retrospective Testing	F (1)			
Remote Testing		IF (2)		
Inspection				
Guideline Review	IF (2)		(3)	M (5)†
Cognitive Walkthrough	IF (2)	F (1)		
Pluralistic Walkthrough	IF (1)			
Heuristic Evaluation	IF (1)			
Perspective-based Inspection	IF (1)			
Feature Inspection	IF (1)			
Formal Usability Inspection	F (1)			
Consistency Inspection	IF (1)			
Standards Inspection	IF (1)			
Inquiry				
Contextual Inquiry	IF (1)			
Field Observation	IF (1)			
Focus Groups	IF (1)			
Interviews	IF (1)			
Surveys	IF (1)			
Questionnaires	IF (1)	IF (2)		
Self-reporting Logs	IF (1)			
Screen Snapshots	IF (1)			
User Feedback	IF (1)			

Table A.1: Automation support for 75 WIMP UE methods (Table 1 of 2). 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). The * for the IFM entry indicates that either formal or informal interface use is required. In addition, a model may be used in the analysis. The † indicates that methods may or may not require a model.

Method Class / Method Type	Automation Type: None	Capture	Analsis	Critique
Analytical Modeling				
GOMS Analysis	M (4)		M (2)	
UIDE Analysis			M (2)	
Cognitive Task Analysis			M (1)	
Task-environment Analysis	M (1)			
Knowledge Analysis	M (2)			
Design Analysis	M (2)			
Programmable User Models			M (1)	
Simulation				
Information Proc. Modeling			M (8)	
Petri Net Modeling			FM (1)	
Genetic Algorithm Modeling		(1)		

Table A.2: Automation support for 75 WIMP UE methods (Table 2 of 2). 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).

A.2 Automation Characteristics of Web UE Methods

Table A.5 summarizes automation characteristics for the 58 UE methods surveyed for Web interfaces. Table A.6 provides descriptions of all method types, and Table A.7 summarizes the number of non-automated and automated capture, analysis, and critique methods surveyed. Three software tools provide automation support for multiple method types: Dome Tree visualization - log file analysis and information scent modeling; WebVIP - performance measurement and remote testing; and WebQuilt - performance measurement, remote testing, and log file analysis.

Method Class Method Type	**Description**
Testing	
Thinking-aloud Protocol	user talks during test
Question-asking Protocol	tester asks user questions
Shadowing Method	expert explains user actions to tester
Coaching Method	user can ask an expert questions
Teaching Method	expert user teaches novice user
Co-discovery Learning	two users collaborate
Performance Measurement	tester or software records usage data during test
Log File Analysis	tester analyzes usage data
Retrospective Testing	tester reviews videotape with user
Remote Testing	tester and user are not co-located during test
Inspection	
Guideline Review	expert checks guideline conformance
Cognitive Walkthrough	expert simulates user's problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies heuristic violations
Perspective-based Inspection	expert conducts narrowly focused heuristic evaluation
Feature Inspection	expert evaluates product features
Formal Usability Inspection	experts conduct formal heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance
Inquiry	
Contextual Inquiry	interviewer questions users in their environment
Field Observation	interviewer observes system use in user's environment
Focus Groups	multiple users participate in a discussion session
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaires	user provides answers to specific questions
Self-reporting Logs	user records UI operations
Screen Snapshots	user captures UI screens
User Feedback	user submits comments
Analytical Modeling	
GOMS Analysis	predict execution and learning time
UIDE Analysis	conduct GOMS analysis within a UIDE
Cognitive Task Analysis	predict usability problems
Task-environment Analysis	assess mapping of user's goals into UI tasks
Knowledge Analysis	predict learnability
Design Analysis	assess design complexity
Programmable User Models	write program that acts like a user
Simulation	
Information Proc. Modeling	mimic user interaction
Petri Net Modeling	mimic user interaction from usage data
Genetic Algorithm Modeling	mimic novice user interaction

Table A.3: Descriptions of the WIMP UE method types depicted in Table A.1.

Methods Surveyed	Automation Type			
	None	Capture	Analysis	Critique
Total	30	5	8	1
Percent	68%	11%	18%	2%

Table A.4: Summary of WIMP UE methods surveyed for each automation type.

Method Class / Method Type	Automation Type			
	None	Capture	Analysis	Critique
Testing				
Thinking-aloud Protocol	F (1)			
Question-asking Protocol	F (1)			
Shadowing Method	F (1)			
Coaching Method	F (1)			
Teaching Method	F (1)			
Co-discovery Learning	F (1)			
Performance Measurement	F (1)	F (4)		
Log File Analysis			IFM (10)	
Retrospective Testing	F (1)			
Remote Testing		IF (3)		
Inspection				
Guideline Review	IF (4)		(5)	(6)
Cognitive Walkthrough	IF (2)			
Pluralistic Walkthrough	IF (1)			
Heuristic Evaluation	IF (1)			
Perspective-based Inspection	IF (1)			
Feature Inspection	IF (1)			
Formal Usability Inspection	F (1)			
Consistency Inspection	IF (1)			
Standards Inspection	IF (1)			
Inquiry				
Contextual Inquiry	IF (1)			
Field Observation	IF (1)			
Focus Groups	IF (1)			
Interviews	IF (1)			
Surveys	IF (1)			
Questionnaires	IF (1)	IF (1)		
Self-reporting Logs	IF (1)			
Screen Snapshots	IF (1)			
User Feedback	IF (1)			
Analytical Modeling				
No Methods Surveyed				
Simulation				
Information Proc. Modeling			M (1)	
Information Scent Modeling		M (1)		

Table A.5: Automation support for 58 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).

Method Class Method Type	**Description**
Testing	
Thinking-aloud Protocol	user talks during test
Question-asking Protocol	tester asks user questions
Shadowing Method	expert explains user actions to tester
Coaching Method	user can ask an expert questions
Teaching Method	expert user teaches novice user
Co-discovery Learning	two users collaborate
Performance Measurement	tester or software records usage data during test
Log File Analysis	tester analyzes usage data
Retrospective Testing	tester reviews videotape with user
Remote Testing	tester and user are not co-located during test
Inspection	
Guideline Review	expert checks guideline conformance
Cognitive Walkthrough	expert simulates user's problem solving
Pluralistic Walkthrough	multiple people conduct cognitive walkthrough
Heuristic Evaluation	expert identifies heuristic violations
Perspective-based Inspection	expert conducts narrowly focused heuristic evaluation
Feature Inspection	expert evaluates product features
Formal Usability Inspection	experts conduct formal heuristic evaluation
Consistency Inspection	expert checks consistency across products
Standards Inspection	expert checks for standards compliance
Inquiry	
Contextual Inquiry	interviewer questions users in their environment
Field Observation	interviewer observes system use in user's environment
Focus Groups	multiple users participate in a discussion session
Interviews	one user participates in a discussion session
Surveys	interviewer asks user specific questions
Questionnaires	user provides answers to specific questions
Self-reporting Logs	user records UI operations
Screen Snapshots	user captures UI screens
User Feedback	user submits comments
Analytical Modeling	
No Methods Surveyed	
Simulation	
Information Proc. Modeling	mimic user interaction
Information Scent Modeling	mimic Web site navigation

Table A.6: Descriptions of the Web UE method types discussed in Table A.5.

Methods Surveyed	**Automation Type**			
	None	**Capture**	**Analysis**	**Critique**
Total	26	5	4	1
Percent	72%	14%	11%	3%

Table A.7: Summary of Web UE methods surveyed for each automation type.

Appendix B

Label Creation Task, Chapter 3

Chapter 3 discusses a label creation task in Microsoft Word 97 and 2000. This section presents the full 55-step label creation task as well as the correlation between this 55-step task sequence and the high-level task presented in Chapter 3.

B.1 Complete Task Sequence

Figures B.1 and B.2 depict the complete task sequence for the label creation task.

B.2 High-Level Task Sequence

Figures B.3 and B.4 depict the correlation between this 55-step task sequence and the high-level task presented in Chapter 3.

1. Select Tools menu
2. Select Envelopes and Labels item
3. Click Options button
4. Select label size from list
5. Click OK button
6. Click New Document button
7. Type address in first label
8. Highlight address
9. Click Center button
10. Select Format menu
11. Select Borders and Shading item
12. Select Box setting
13. Click OK button
14. Click on right mouse button while over highlighted text
15. Select Copy option
16. Move cursor to second label (top middle)
17. Click right mouse button
18. Select Paste option
19. Highlight address
20. Select Format menu
21. Select Borders and Shading item
22. Select Box setting
23. Click OK button
24. Move cursor to third label (top right)
25. Click right mouse button
26. Select Paste option
27. Highlight address

Figure B.1: Complete task sequence for creating address labels (six on a sheet with a square border around each label) in Microsoft Word 97 and 2000 (Figure 1 of 2).

28. Select Format menu

29. Select Borders and Shading item

30. Select Box setting

31. Click OK button

32. Move cursor to fourth label (bottom right)

33. Click right mouse button

34. Select Paste option

35. Highlight address

36. Select Format menu

37. Select Borders and Shading item

38. Select Box setting

39. Click OK button

40. Move cursor to fifth label (bottom middle)

41. Click right mouse button

42. Select Paste option

43. Highlight address

44. Select Format menu

45. Select Borders and Shading item

46. Select Box setting

47. Click OK button

48. Move cursor to sixth label (bottom left)

49. Click right mouse button

50. Select Paste option

51. Highlight address

52. Select Format menu

53. Select Borders and Shading item

54. Select Box setting

55. Click OK button

Figure B.2: Complete task sequence for creating address labels (six on a sheet with a square border around each label) in Microsoft Word 97 and 2000 (Figure 2 of 2).

1. Select Tools menu **(1. Change document to label format)**
2. Select Envelopes and Labels item
3. Click Options button
4. Select label size from list
5. Click OK button
6. Click New Document button
7. Type address in first label **(2. Type address)**
8. Highlight address **(3. Center address)**
9. Click Center button
10. Select Format menu **(4. Add border)**
11. Select Borders and Shading item
12. Select Box setting
13. Click OK button
14. Click on right mouse button while over highlighted text **(5. Copy address)**
15. Select Copy option
16. Move cursor to second label (top middle) **(6. Move cursor to next label)**
17. Click right mouse button **(7. Paste address)**
18. Select Paste option
19. Highlight address **(8. Select address)**
20. Select Format menu **(9. Add border)**
21. Select Borders and Shading item
22. Select Box setting
23. Click OK button
24. Move cursor to third label (top right) **(10. Move cursor to next label)**
25. Click right mouse button **(11. Paste address)**
26. Select Paste option
27. Highlight address **(12. Select address)**

Figure B.3: High-level task sequence for creating address labels (bold entries on the right, Figure 1 of 2).

28. Select Format menu **(13. Add border)**

29. Select Borders and Shading item

30. Select Box setting

31. Click OK button

32. Move cursor to fourth label (bottom right) **(14. Move cursor to next label)**

33. Click right mouse button **(15. Paste address)**

34. Select Paste option

35. Highlight address **(16. Select address)**

36. Select Format menu **(17. Add border)**

37. Select Borders and Shading item

38. Select Box setting

39. Click OK button

40. Move cursor to fifth label (bottom middle) **(18. Move cursor to next label)**

41. Click right mouse button **(19. Paste address)**

42. Select Paste option

43. Highlight address **(20. Select address)**

44. Select Format menu **(21. Add border)**

45. Select Borders and Shading item

46. Select Box setting

47. Click OK button

48. Move cursor to sixth label (bottom left) **(22. Move cursor to next label)**

49. Click right mouse button **(23. Paste address)**

50. Select Paste option

51. Highlight address **(24. Select address)**

52. Select Format menu **(25. Add border)**

53. Select Borders and Shading item

54. Select Box setting

55. Click OK button

Figure B.4: High-level task sequence for creating address labels (bold entries on the right, Figure 2 of 2).

Appendix C

Benchmarking Tools and Web Interface Measures, Chapters 4 and 5

C.1 Benchmarking Tools

The Site Crawler, Metrics Computation, and Analysis Tools described in Chapter 4 are available for online use via the WebTango Project Tools page (http://webtango.berkeley.edu/tools/).

C.2 Web Interface Measures

An interactive appendix was developed to illustrate all of the Web interface measures discussed in Chapter 5. This appendix can be loaded from the CDROM accompanying this dissertation (or from the Computer Science Division) by opening the file appc/web/index.html in a Web browser or appc/appc.ppt in Microsoft PowerPoint. The HTML version of the appendix is also available at http://webtango.berkeley.edu/tools/metrics/web/index.html; the PowerPoint version is available at http://webtango.berkeley.edu/tools/metrics/appc.ppt.

This appendix is best viewed in Microsoft PowerPoint because there is some strange wrapping in the HTML version generated from PowerPoint.

Appendix D

Comparison of Original and Modified Web Pages, Chapters 8 and 9

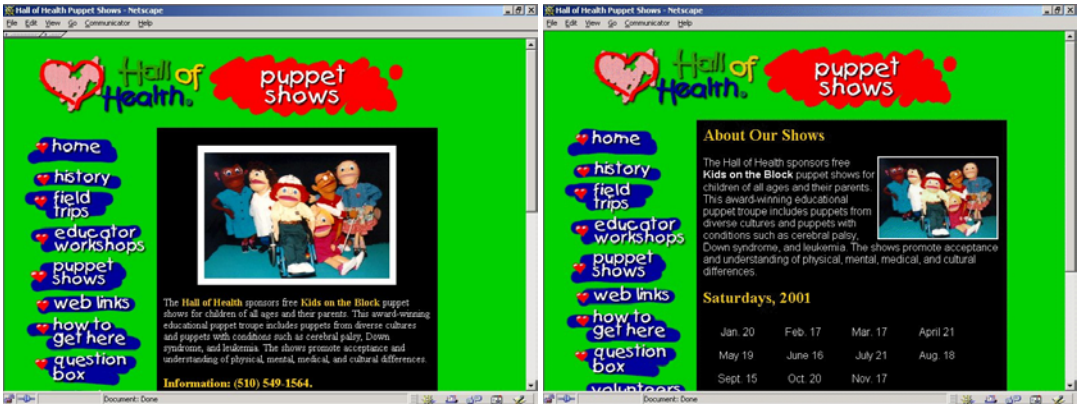
D.1 Web Pages from the Example Assessment

Chapter 8 summarized the assessment of an example health education site. It presented images of three example pages from the site as well as images of the corresponding modified pages. Figure D.1 provides a side-by-side comparison of the original home, link, and content pages and the pages modified based on the profiles developed in Chapter 6.

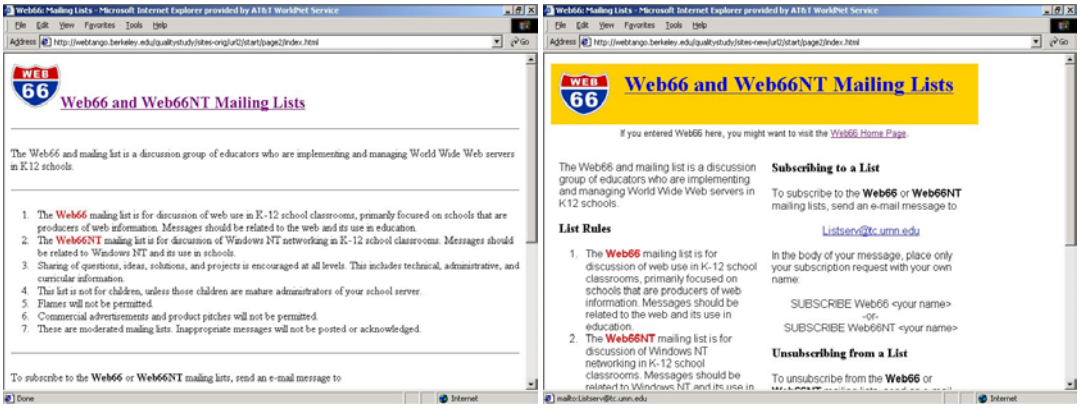
D.2 Web Pages from the Profile Evaluation Study

Chapter 9 described a study that focused on assessing whether pages and sites (including the example site from Chapter 8) modified based on the profiles were preferred over the original pages and sites. It presented images of example pages from five sites as well as images of the corresponding modified pages. Figures D.2 and D.3 provide side-by-side comparisons of the original and modified pages.

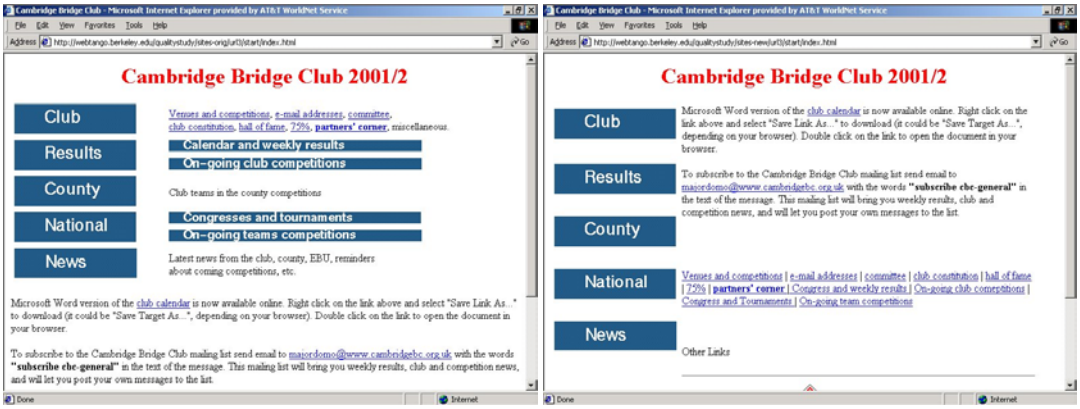
Figure D.1: Home (top), link (middle), and content (bottom) pages taken from the example health education site discussed in Chapter 8. Many of the changes in the modified pages are not visible. These include a set of text links at the bottom of the page that mirror the graphical links, removal of colored and italicized body text words, and addition of an accent color.



Original Example Page (Site 1) Modified Example Page (Site 1)

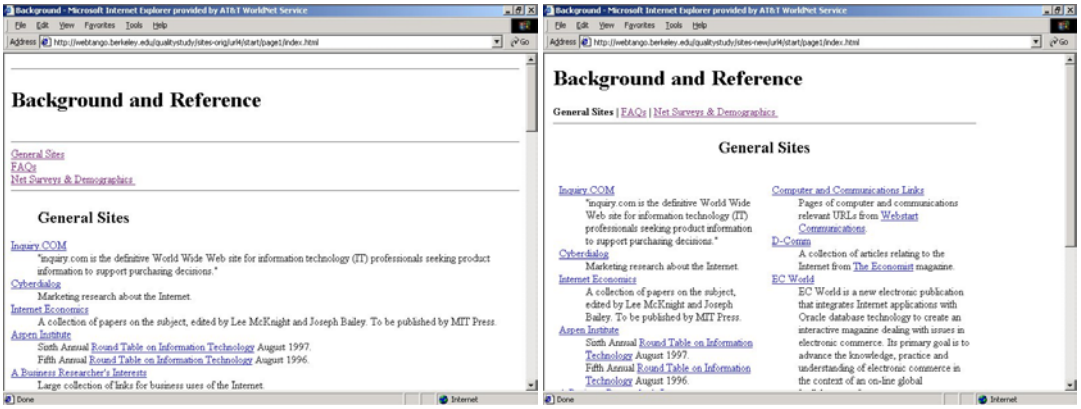


Original Example Page (Site 2) Modified Example Page (Site 2)

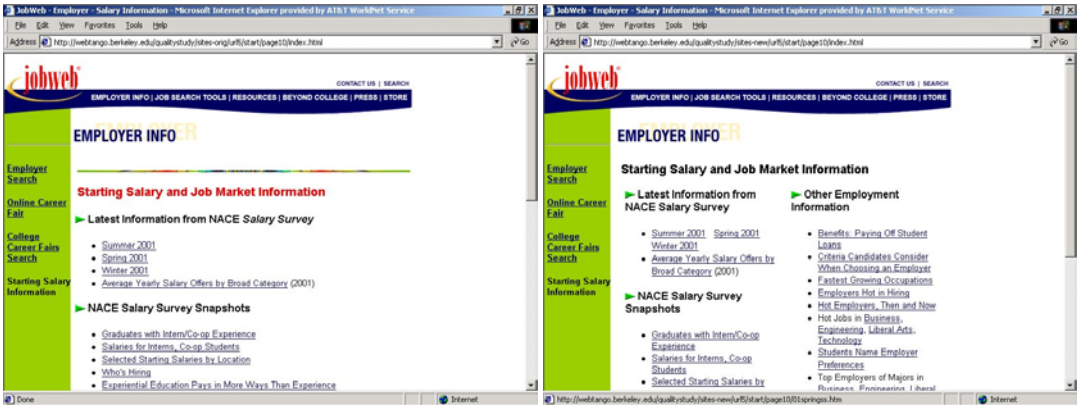


Original Example Page (Site 3) Modified Example Page (Site 3)

Figure D.2: Example pages from study sites 1 (top), 2 (middle), and 3 (bottom). Some of the changes in the modified pages are not visible.



Original Example Page (Site 4) Modified Example Page (Site 4)



Original Example Page (Site 5) Modified Example Page (Site 5)

Figure D.3: Example pages from study sites 4 (top) and 5 (bottom). Some of the changes in the modified pages are not visible.