Behavior Research Methods, Instruments, & Computers
2003, 35 (2), 194-201

ARTICLES FROM THE SCIP CONFERENCE

Uzilla: A new tool for Web usability testing

ANDY EDMONDS
Clemson University, Clemson, South Carolina

Web usability testing and research presents challenges for accurate data collection. An instrumented browser solution, Uzilla, is compared with existing solutions, and its contributions to usability testing practice are noted. Uzilla implements a client–server architecture based on the open source Mozilla browser. Instrumentation of the browser facilitates the evaluation of Web sites and applications inside and outside of the laboratory. An integrated data collection and analysis server application decreases the effort required to understand test results and facilitates iterative testing.

More people use Web browsers than any other class of desktop software. This situation creates a previously unparalleled level of user experience in a software niche. It also creates unique challenges as a Web site or application becomes a hybrid of the design of the site and of the browser. This paper introduces a system for assessing user behavior within this environment, aimed at optimizing the process of usability testing and enabling basic research in this area.

Usability testing has arisen in the last 10 years as a beneficial practice in software development (Dumas, 2002). Key attributes of a usability test include the recruitment of representative users and the assignment of top priority tasks. Additional attributes commonly associated with usability testing are the collection of verbal protocols and the use of surveying techniques to assess user satisfaction.

Tools for Web site usability testing need to log actions both within the site and within the browser. Uzilla is an instrumented browser and data server that permits both efficient testing and precise logging. By automating the presentation of instructions and surveys and providing extensive analysis tools, it facilitates the broad use of usability testing.

Uzilla works in the laboratory, at a user's desktop, and within the rapidly growing practice of remote usability testing, a technique that will be reviewed later in this paper.

The case study mentioned in this paper was conducted by Kranti Dugiraala, Pallavi Dharwada, Andy Edmonds, Jonathan Johnson, Deepthi Nalanagula, and Sajay Sadasivan under the supervision of Andrew Duchowski. Thanks Leo Gugerty, Steve Davis, Rick Tyrrell for valuable feedback on this work. Uzilla.net is available from Uzilla, LLC as a hosted service, licensed on a per project basis. A 50% academic discount is available from the prices posted on www.uzilla.net. No cost licensing is available for student projects and classroom use. Correspondence should be addressed to A. Edmonds, 110 Gregory St., Clemson, SC 29631(e-mail: andy@uzilla.net).

An application server and database drive a Web application for test design, data aggregation, and reporting. By logging URLs visited, Uzilla eliminates the need to hand-code video to segment user observation by page, as was reported in a recent Web study (Goldberg, Stimson, Lewenstein, Scott, & Wichansky, 2002). Uzilla efficiently stores the rapid data accumulation in a format convenient for analysis.

The results generated from a single test include quantitative and qualitative measurements. Several techniques for problem identification have been implemented, including error page detection and measures of nonoptimality. However, the results must be analyzed by a skilled usability analyst familiar with the Web site studied to identify which behavior and survey feedback raises problems. The quantitative results are particularly useful in summative, or late stage, software development, whereas the identification of errors and survey feedback may be useful with early prototypes.

The Uzilla architecture enables comparisons across tests of product design iterations. Iterative testing and the monitoring of improvement has the potential to greatly aid the evolution of Web applications. Iterative testing may also provide a methodology for assessing rate of learning and memorability, two recognized but typically difficult to measure aspects of software usability.

Finally, Uzilla enables experimentation in browser-based interface design and behavior. The availability of browser instrumentation has already made significant contributions to the understanding of Web behavior, the design of Web sites, and even the design of browsers.

In the following, the Uzilla application will be described, and a case study will be presented showing its use in an experimental evaluation of design alternatives. Uzilla then will be compared with other methods of facilitating Web usability testing, and a brief survey of testing methodology and related research will be offered.

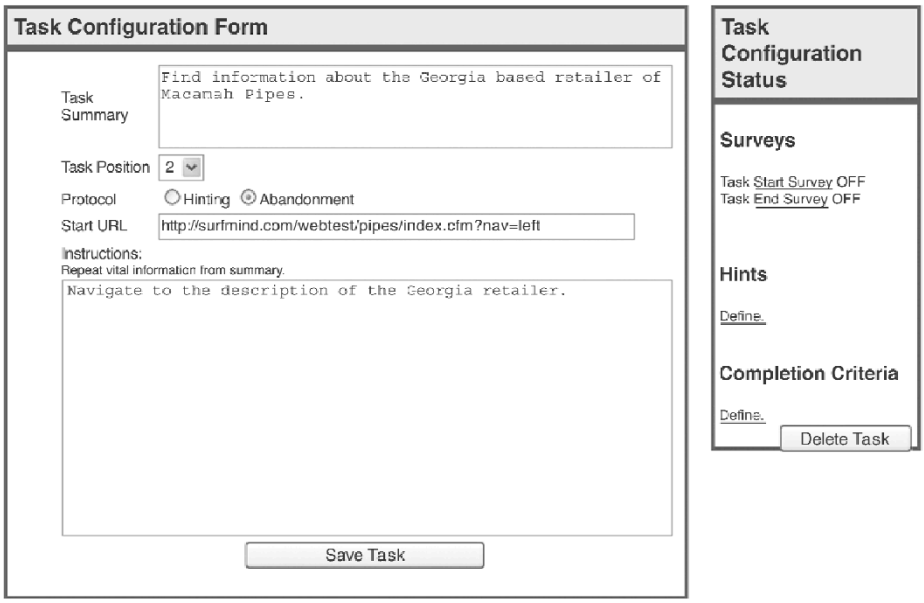

Figure 1. The task design screen supports the specification of task instructions, hints, completion criteria, and surveys.

Uzilla Specification

Uzilla combines an instrumented browser and a Web application for usability test design and reporting. The system uses a traditional three-tier architecture with a browser client, a Web application server, and a database. The usability test design application allows the experimenter to specify instructions, surveys, and task definitions.

Task definition includes specifying instructions, hints, and surveys. The hinting protocol is derived from the Common Industry Format (CIF) from NIST (1999a). Surveys may be used prior to task assignment to assess expectations or afterward to assess task completion and satisfaction. Although individual task executions may be manually scored, a text-matching system for the user's final pageview in the task enables automatic scoring. Figure 1 shows the task design screen.

Uzilla's browser is based on the open source Mozilla browser, the basis for Netscape 7. It utilizes a custom toolbar to present usability test instructions and the browser to conduct surveys (see Figure 2). Extensions to the Mozilla in JavaScript and XPCOM support logging network, mouse, and keyboarding events.[1]

When a test is begun, the browser retrieves the test definition from the application server and walks the user through an initial survey and subsequent task assignments. Data are logged with every page transition via HTTP to a Web server and are then transferred to a set of tables in a database. The transmission of data to a central repository via the Internet enables remote usability testing. Uzilla could be extended to enable real-time communication, enabling playback of user activity similar to screen-sharing programs, but would likely be blocked by many corporate firewalls.

Uzilla logs user activity, such as typing, scrolling, and mouse movement. Mouse motion is compressed to vectors with a directional summary for simplified aggregation, using an open source gesture recognition algorithm (Optimoz, 2002). For each page view, Uzilla logs the time of page html content load, the time of asset (e.g., images, scripts, Flash movies) load completion, and the point at which the next page is requested. Uzilla effectively collects data from pages with frames, as well as multiple window-browsing sessions.

Mouse clicks are logged with an identifying path from within the document object model (DOM) of the html page or Mozilla browser interface. For a form submit button, this might produce a path of “Create Report” of type submit button in form “selectDate.” For links, the link text, page location, and the URL are logged, allowing the discrimination of potentially redundant links on a page.

Figure 2. The Uzilla toolbar presents the user’s current task and options.

The DOM path technique is also used to identify browser UI operations in the Mozilla cross platform UI. Figure 3 shows the DOM view of the browser toolbar. The highlighted row is the back button. Events, captured via JavaScript and the W3C event model, are distinguished by the ID of the source element in the DOM. Uzilla has a behavior report that analyzes the event stream for any of several ways in which a back button may be invoked.

Figure 4 shows a text rendering of a page event trace in which a user navigates to a search input box on google.com, types a query, and moves the mouse to click the submit button. Interactions with a form element are stored with a string representing the DOM path of the element and relevant identifiers in that path. Interactions with specific form elements may be identified across a collection of users by a database query searching for the form element ID or name.

Report generation follows the requirements of the CIF from NIST (1999a). The key measures of success rate and time to complete are computed automatically, along with performance measures of mouse miles and click count. Reporting is grouped into task, page, user, survey, and behavior functions (see Figure 5). Export facilities are provided for data transfer to dedicated statistical analysis software. The reporting module offers typical graphs in addition to Scalable Vector Graphic visualizations of user activity.

In Hilbert and Redmiles's (2000) taxonomy of possible support systems for event-logging software, Uzilla provides integrated evaluation support and event stream transformations through aggregations of task time and success rate, as well as page reports of mouse, keyboard, and form details. Future work will add synchronization with usability lab video systems, administrator notes, and eye-tracking data.

Hilbert and Redmiles (2000) identified the association of event traces with system state as a key challenge in the use of event logging in usability evaluation. On the Web, this problem is simplified, since the page serves as a useful state indicator. Although not totally sufficient in all cases to identify the system status, the current page, combined with the recent viewing history and form activity log, tends to uniquely identify system status.

An important aspect to the use of URLs as indentifiers of system status and as a level of usability analysis is the use of dynamic pages whose content may be governed by URL parameters. Uzilla stores pages and URL parameter strings in separate database tables, allowing reporting at either level. For example, this would allow one to isolate behavior for a product page template or for a specific product.

Although sequence analysis is a complex process, limited support for comparison of user paths with the optimal is available in Uzilla. Optimal paths are simply a flagged user test session. A visualization shows the deviation of observed paths from optimal, thus informing on suboptimal behavior. Uzilla could be extended to include form behavior in the optimality analysis and to allow the specification of multiple paths as optimal.

Case Study

An experimental study of left versus right navigation was conducted at Clemson University in late 2002 involving two similar sites with either right or left navigation. Uzilla was used to capture the behavior and survey the responses of 8 participants in a within-subjects design. The experiment counterbalanced the two site designs with navigation placement, and the protocol emulated a usability test with users assigned a set of three information-finding tasks with each site.

Uzilla's iterative comparison facilities are based on hereditary relationships in the task definition and were not sufficient to generate all of the necessary comparisons. However, the system was easily extended so as to pool the equivalent experimental conditions and use the built-in reporting mechanisms. The data were then extracted for analysis in a dedicated statistical analysis package.

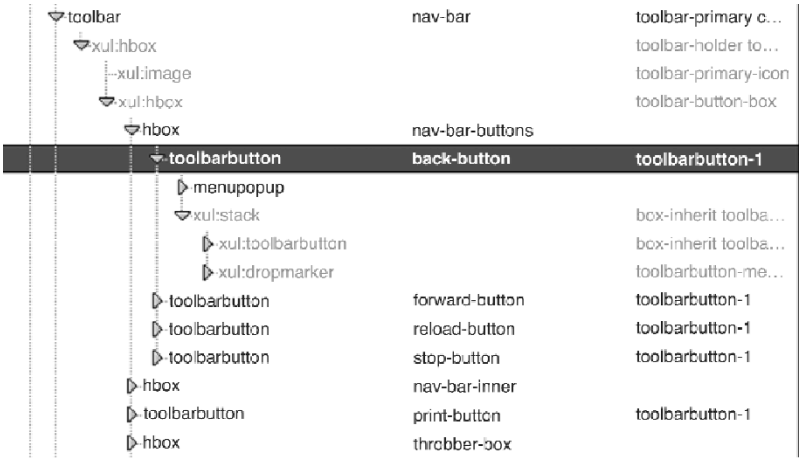

Figure 3. The Uzilla page report detailing the timing of load events and the onset of user activity for a particular page.

x	y	op	op detail	depth	time	Dompath
		Request			0ms	
		ContentLoad			160ms	
		Network End			16233ms	
354	131	0	vstart	0px	4045ms	'na'
356	36	DUD	vend	0px	4316ms	'na'
362	44	0	vstart	0px	4887ms	'na'
588	168	R	vend	0px	5087ms	'na'
578	182	0	mousedown	0px	7480ms	'NAME:q VALUE:;NAMEf ;'
578	182	0	mouseup	0px	7621ms	'NAME:q VALUE:;NAMEf ;'
0	0	instrument	key	0px	11035ms	'NAME:q VALUE:;NAMEf ;'
0	0	ed browse	key	0px	12628ms	'NAME:q VALUE:;NAMEf ;'
0	0	r<enter>	key	0px	14190ms	'NAME:q VALUE:;NAMEf ;'
		Unload			16373ms	

Figure 4. The event trace for a page showing load timing, mouse movement vectors, keyboarding events, and the page unload.

Two of the tasks required an average of three page views, whereas the third required twice as many. For brevity, only a summary of the lengthier task will be presented. The time on task and error measures revealed that the mean time for the left navigation condition was 42 sec, whereas the right hand navigation required 62 sec. The measurement of mouse travel distance, referred to as mouse miles, was used to measure the amount of work required by the participant in the different designs.

A comparison of the left with the right condition showed only a small difference in pixel distance traveled but a large difference in the ratio of horizontal to vertical travel, with the users spending much more time in horizontal mouse movement in the right-hand condition. Subjectively, the users showed no preference for either navigation scheme.

Methods in Usability Measurement

Scientific understanding of usability testing is immature and inconclusive. Such issues as administrator reliability, participant bias by an administrator, effects of different verbal protocol instructions, and many others are still being researched and debated. The reader is encouraged to read Dumas (2002) for a more detailed treatment of the relevant research. NIST's CIF format is an attempt to standardize both the usability process and the reporting of re-

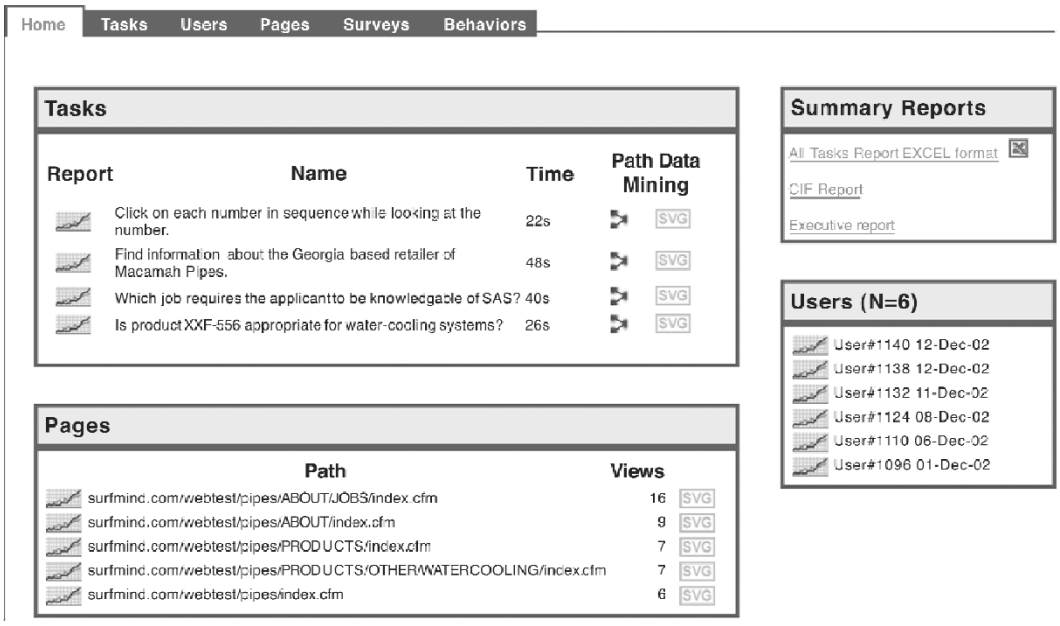

Figure 5. The Uzilla.net analysis homepage groups data analysis functions by task, page, user, survey, and behaviors.

sults. It provides a set of measurements that enable a usability professional to determine a general measure of the usability of a system. In this case, quantitative data is very helpful. In formative evaluations, where a skilled administrator may vary the questioning of a participant according to his or her individual experience with a system, quantitative measures, such as time on task, may be less useful.

Uzilla reports traditional measures of usability, as proposed in the CIF, of task success, task time, and subjective measures of satisfaction. An optimal path feature provides the basis for measuring deviations in terms of time, number of page views, number of clicks, and mouse miles. A related protocol for administrator scoring of user behavior has recently been proposed by Faulkner (2002) as a methodology for improving the scoring process and increasing the reliability of test-logging processes.

Quantitative measures collected in a robust and automated way, as with Uzilla, may also extend the use of baseline and iterative usability evaluation. Dumas (2002) examined the use of subjective surveys employed over the course of a product's development to chart progress toward usability goals. Uzilla enables the comparison of performance measures across product versions. This is especially critical in Web sites and applications, where changes often occur much more rapidly due to volatile content and the ease of updating a Web server.

Performance measures also serve well in competitive evaluations in which designs are pitted against one another. Such investigations are challenging due to the need to test more participants. Uzilla's automation may provide the opportunity to empirically investigate alternate designs by reducing the overall time required. Another complication in competitive evaluations occurs when a comparison is desired between a site and an actual competitor in the marketplace and it is impossible to use site instrumentation methods. Instrumenting the browser enables testing of any site on the Internet.

The common practice of testing small numbers of users (fewer than 10, typically) is generally regarded as sufficient to reveal the key problems, with additional tests resulting in diminishing returns (Virzi, 1992). Although this result is not unequivocal (Dumas, 2002), there are additional motivations for testing more users beyond increasing the percentage of problems revealed. Problem severity ratings are dependent on the skill of the tester, and evaluating problems is an intuitive process. Through the testing of additional participants, more reliable problem frequency measures may be obtained. This is compatible with the measure recommended by Rubin (1994), in which expert estimations of severity are multiplied by the number of users who experienced the problem.

Remote usability testing. In remote usability testing, participants are typically at their own workstations. In some scenarios, a voice connection is made to a test administrator; usability professionals have also reported using screen-sharing programs to view a remote participant's screen (Hammontree, Weiler, & Nayak, 1994). Tullis, Fleischman, McNulty, Cianchette, and Bergel (2002) compared two tests in lab and remote settings with samples of 8 lab users and 29 remote users in Experiment 1 and 8 and 88, respectively, in Experiment 2. With only limited instrumentation, they discovered that both methods revealed the same core problems and resulted in similar task time and success measures. Each protocol revealed some problems that the other did not, with the lab testing revealing more unique problems in Experiment 1, given a sample that was five times larger in the remote test, and with remote testing producing more unique problems in Experiment 2 with an 11 times larger sample size.

The technological solution used by Tullis et al. (2002) did not allow them to circumvent browser security restrictions, preventing recording of the page paths and intrapage activity. Because of this, their comparison may have underevaluated the potential of remote testing. The problems that were uniquely identified were summarized as, "We saw evidence of certain kinds of user behaviors in the lab (e.g., excessive scrolling, failure to see certain elements on the screen at first) that were less likely to be captured in the remote tests" (p. 5). Excessive scrolling is recorded in Uzilla, and a failure to see a critical element might be identified through mousing behavior.

Tullis et al.'s (2002) study suggests that remote testing can be a fruitful way to do usability valuation and that the potential for increased sample sizes can produce more robust results. Their results also suggest that comment data is critical. Ebling and John (2000) analyzed the source of usability problem identifications from quantitative versus protocol data in a single usability test. Protocol data was found to contribute uniquely and to the greatest extent to problem identification, although quantitative measures replicated Tullis et al.'s finding of a high detection rate for major issues. Effective solicitation of user comments or verbal protocol in remote testing is a key challenge for its success.

Future extensions of Uzilla may add the capability to record participant verbalizations and synchronize them with the browser event stream. In its current form, combining Uzilla with lab technology or telephony is the best approach to capturing verbalizations. Uzilla does provide a comment option for users to type free form impressions. Participant note-taking is way to capture critical user protocol data. This is enabled in a related protocol, termed desktop usability testing, in which a test administrator brings the lab to the user with Uzilla. This preserves many of the benefits in remote testing that lead to a greater number of participants and increases the external validity of the situation by testing the user in his or her natural environment.

Uzilla addresses remote and desktop testing through a downloadable installer or auto-run CD that requires no installation process. The system does not address sampling site approaches in which site visitors are recruited to participate in a test protocol motivated with a gift certificate or prize lottery. This practice is supported by a number of vendors but is outside the scope of usability testing. A related nonusability test scenario is pilot-testing of alternate designs on a selection of live Web site users. Posts on the ACM CHI mailing list have reported that Google has used

this technique, much to the chagrin of curious users hoping to see the new version. These types of data collection are likely to complement, but not replace, more traditional usability testing protocols.

Instrumentation in Software for Usability

Instrumentation of software products for measuring usability is a long-standing technique (Good, 1985). A key critique of instrumentation systems has been the lack of tools for analyzing the extensive accumulated data (Hilbert & Redmiles, 2000). The use of a relational database with a well-designed schema for the usability test data allows extraction of statistics at different granularities. Structured Query Language (SQL) operations include not only data selection, but also aggregation operations, such as averaging, computing standard deviations, and computing differences. For instance, time on page may be computed in the database layer by selecting a "time on page" variable as end time minus start time from the page view table. Thus, Uzilla provides critical summary statistics, such as time on task, nonoptimal page views, and aggregations of survey responses. A key design criterion was to match the output requirements of the NIST (1999a) CIF.

Although the utility of low-level data, such as keypresses and mouse vectors, has not generally been productive in usability analysis, there are a number of applications in the Web space for which these data are likely to provide important information. There is a continuum of Web sites and applications from brochure-ware to data entry systems, with the utility of low-level data increasing as the site is used more frequently. In a call center Web-based application, for example, every inefficiency in the system is crucial. Forms in all types of sites are likely to be improved through careful analysis of keystroke level data, particularly given the critical importance forms play in conducting e-commerce and connecting the customer with a business.

Previously Implemented Systems

Hilbert and Redmiles (2000) have offered a complete review of event-logging systems for extracting usability information. The core observation techniques for the Web are server logs, instrumented sites, proxies, and instrumented browsers. Although Web server log files may record data that are useful for capturing Web behavior in a usability test, the data are high level and are subject to significant data loss through browser caching mechanisms. Of particular note is the use of the back button to return to the homepage from an errant first click. The second visit to the homepage may not be recorded in the Web server log, due to browser caching. This problem can be addressed programmatically by parsing every session trail and computing missing pages (Hong, Heer, Waterson, & Landay, 2003); however, popular log file analysis systems typically do not do this form of analysis.

A separate class of systems instrument a Web site with code that records user activity. WebVIP (NIST, 1999b), WET (Etgen & Cantor, 1999), and Lucidity (Edmonds, 2001) are noncommercial examples of such systems and require that a site be instrumented with JavaScript includes.[2] These systems enable accurate observation of page views, independent of caching, but cannot log browser UI events such as the back button and the entry or modification of http addresses in the address bar.

There are several browsers that log interface operations. They include WebLogger (Reeder, Pirolli, & Card, 2001), WebTracker (Choo, Detlor, & Turnbull, 2000a), and ErgoBrowser (Ergosoft Laboratories, n.d.). These systems typically embed a Microsoft Web browser ActiveX control inside a custom interface. In contrast, Uzilla uses the Mozilla browser engine to enable logging of user activity within the page, as well as browser level operations, such as the back button.

Proxies intercept requests from a browser, providing the opportunity to log the page requests and potentially augment the content with instrumentation. WebQuilt (Hong & Landay, 2001) is a freely available system of this type. Proxies enable the testing of sites that a tester may not have the ability to instrument but require additional computing resources and may encounter difficulty with JavaScript-based navigation systems. Finally, all browsers store bookmarks and history information in the file system, and research has been conducted using solely these sources (e.g., Cockburn & McKenzie, 2001). A promising area of research is to study the way Web browsing is related to history and bookmark data stores, an inquiry requiring access to the browser internal data structures.

Although Uzilla's data-logging facilities are on par with previous efforts in this area, a key innovation is the addition of support for the design and running of usability tests and an integrated application to help analyze the results. The thick client, or browser-based, approach does not offer the ability to sample Web site visitors in the field, as with a site instrumentation approach, but the likelihood is that individuals in this scenario would rarely consent to an hour or longer process, as is typical in usability testing. The payoff is complete control over the user environment, browser preference settings, support for testing of any site on the Internet, integrated support for the testing process, and the ability to log operations outside of the Web page but within the browser UI.

Related Work and Research Opportunities

Educational applications.Uzilla may be of use in human–computer interaction instruction as a way to teach the process of usability testing. The design application removes legwork by minimizing material preparation time. The system's reporting makes task and page summaries easy to obtain, freeing a test administrator from significant data aggregation chores. The improved efficiency might allow students to focus more on the important decisions and less on the paper trail in usability test preparation.

Browser-based behavior. Much of the published work on Web-browsing behavior has focused on multisite, multisession browsing activity (Choo, Detlor, & Turnbull, 2000b). Catledge and Pitkow (1995), with an early browser

instrumentation, found that 40% of all browser UI operations were back button clicks. In addition to facilitating replication of previous studies, low-level data will help characterize search on the page versus learned link seeking. Browser instrumentation is a critical enabler of an understanding of site navigation triggers—for example, bookmarks, search engines, and URL entry.

Although Uzilla is focused on protocols in which the user is given a specific goal to accomplish, aggregated data from numerous usability tests could provide additional insight into general Web behavior. Normative statistics on time to execute and error rate with common form elements could better inform site designers in making design decisions.

Mousing behavior on the Web is largely unstudied. To our knowledge, there is only one recent study of mousing behavior (Lockerd & Mueller, 2001). Important observation opportunities include keyboard versus mouse navigation, activity during page views not in path to the eventual exit, and trends in mouse positioning at page start and while scrolling. Potentially, the effect of commonalities in site designs will predict mouse positions and provide information on optimal site design. This is also a particularly fruitful area of investigation, since computational cognitive models are beginning to address motor movement (Byrne, 2001).

Browser instrumentation and development has led to the understanding of revisitation patterns (Tauscher & Greenberg, 1997) and the development of more robust history and bookmarking facilities (Cockburn, Greenberg, Jones, McKenzie, & Moyle, 2003). Other work uses logging of page activity to infer user interest for personalization (Claypool, Le, Waseda, & Brown, 2001). In addition to advancing the success and ease of deployment of usability testing, instrumentation solutions may facilitate additional insight into Web behavior. Significant work exists on the nature of Web browsing and searching (Choo, Detlor, & Turnbull, 2000a) and the use of browser controls (Cockburn & McKenzie, 2001).

Although Uzilla offers support for such investigations, the Mozilla browser presents additional opportunities for experimentation in browser UIs, agents for Web browsing, and support for Web-based tasks. Customization is possible with lighter weight tools than those required for customizing Microsoft Internet Explorer, and the open source nature of Mozilla means that a greater amount of the browser's functionality is customizable. This enables the exploration of alternative browser interface designs, as well as browsing adjuncts. These capabilities can be augmented with Uzilla's data-logging system to provide built-in instrumentation.

The Mozilla codebase is also a robust platform for the development of nonbrowser applications. The entire interface of Mozilla is constructed in an HTML-like tagset that includes richer controls than those available in HTML. Trees, drag and drop, and data binding are notable features of Mozilla as a platform, with the added benefit of Windows, Macintosh, and Linux compatibility. Uzilla's DOM-based event identification would serve well in this scenario, although depending on design, the page-based analysis might not serve to identify system state as well as it does in browsing tasks. The recent maturation of the XPath (W3C, 1999) standard provides a longer term, standards-based approach to identifying elements in a potentially changing document.

REFERENCES

Byrne, M. D. (2001). *A quantitative simulation framework for human factors engineering: ACT–R/PM*. Paper presented at Human. Systems 2001, Houston.

Catledge, L. D., & Pitkow, J. E. (1995). Characterizing browsing strategies in the World-Wide Web. In *Proceedings of the 3rd International World Wide Web Conference* (pp. 1065-1073). Darmstadt, Germany.

Choo, C., Detlor, D., & Turnbull, D. (2000a). *Web work:Information seeking and knowledge work on the World Wide Web*. Dordrecht: Kluwer.

Choo, C., Detlor, B., & Turnbull, D. (2000b). Working the Web: An empirical model of Web use. In *Proceedings of Hawaii International Conference on Systems Science 33*. Maui, HI: IEEE.

Claypool, M., Le, P., Waseda, M., & Brown, D. (2001). Implicit interest indicators. In *Proceedings of ACM Intelligent User Interfaces Conference (IUI)* (pp. 33-40). New York: ACM Press.

Cockburn, A., Greenberg, S., Jones, S., McKenzie, B., & Moyle, M. (2003). Improving Web page revisitation: Analysis, design, and evaluation. *Information Technology & Society*, **3**, 159-183.

Cockburn, A., & McKenzie, B. (2001). What do Web users do? An empirical analysis of Web use. *International Journal of Human–Computer Studies*, **54**, 903-922.

Dumas, J. (2002). User-based evaluations. In J. Jacko & A. Sears (Eds.), *The human–computer interaction handbook* (pp. 1093-1117). NJ: Erlbaum.

Ebling, M. R., & John, B. E. (2000). On the contributions of different empirical data in usability testing. In *Conference proceedings on designing interactive systems* (pp. 289-296). New York: ACM Press.

Edmonds, A. (2001). *Lucidity*. Retrieved March 3, 2003 from http://sourceforge.net/projects/lucidity/.

Ergosoft Laboratories (n.d.). *ErgoBrowser*. Retrieved March 3, 2002 from http://www.ergolabs.com/resources.htm.

Etgen, M., & Cantor, J. (1999). What does getting WET (Web event-logging tool) mean for Web usability? In *Fifth Human Factors and the Web Conference Proceedings*. Retrieved March 3, 2002 from http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/.

Faulkner, L. L. (2002). *Quantifying and analyzing human behavior during usability testing: Cross user analysis and the optimal path test method*. Manuscript submitted for publication.

Goldberg, J. H., Stimson, M. J., Lewenstein, M., Scott, N., & Wichansky, A. M. (2002). Eye tracking in Web search tasks: Design implications. In *Proceedings of the Eye Tracking Research and Applications Symposium* (ETRA) (pp. 51-58). New York: ACM Press.

Good, M. (1985). The use of logging data in the design of a new text editor. In *Proceedings of CHI '85 Human Factors in Computing Systems* (pp. 93-97). New York: ACM Press. New York.

Hammontree, M., Weiler, P., & Nayak, M. (1994). Remote usability testing. *Interact*, **1**, 21-24.

Hilbert, D., & Redmiles, D. (2000). Extracting usability information from user interface events. *ACM Computing Surveys*, **32**, 384-421.

Hong, J. I., Heer, J., Waterson, S., & Landay, J. A. (2003). WebQuilt: A proxy-based approach to remote Web usability testing. *ACM Transactions on Information Systems*, **19**, 263-285.

Hong, J. I., & Landay, J. A. (2001). WebQuilt: A framework for capturing and visualizing the Web experience. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)* (pp. 718-724). Hong Kong: Retrieved March 3, 2002 from http://www10.org/cdrom/.

Lockerd, A., & Mueller, F. (2001). Cheese: Tracking mouse movements on Websites: A tool for user modeling. In *Computer Human In-*

teraction Conference Proceedings (pp. 279-280). New York: ACM Press.

NIST (1999a). *The IUSR project: Industry usability report.* Retrieved March 3, 2002 from http://zing.ncsl.nist.gov/iusr/documents/WhitePaper.html.

NIST (1999b). *WebVIP: Overview*. Retrieved March 3, 2002 from http://zing.ncsl.nist.gov/WebTools/WebVIP/overview.html.

Optimoz (2002). *CVS tag 0.3.4.* Retrieved March 3, 2002 from http://optimoz.mozdev.org/gestures/.

Reeder, R. W., Pirolli, P., & Card, S. K. (2001). *WebLogger: A data collection tool for Web-use studies* (UIR Technical Reports UIR-R-2000-6). Xerox PARC.

Rubin, J. (1994). *Handbook of usability testing*. New York: Wiley.

Tauscher, L. & Greenberg, S. (1997). How people revisit Web pages: Empirical findings and implications for the design of history systems. *International Journal of Human–Computer Studies*, **47**, 97-137.

Tullis, T., Fleischman, S., McNulty, M., Cianchette, C., & Bergel, M. (2002). *An empirical comparison of lab and remote usability testing of Web sites*. Paper presented at the Usability Professionals Conference, Orlando, FL.

Virzi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, **34**, 457-468.

W3C (1999). *XPath.* Retrieved March 03, 2003 from http://www.w3.org/TR/xpath.

NOTES

1. Accuracy is limited by PC keyboard refresh and variance in the synchronization of presentation technology and screen refresh.

2. JavaScript includes circumvent the page-caching issue by adding an additional http transaction for every page load.

(Manuscript received November 22, 2002;
revision accepted for publication March 1, 2003.)