Behavior Research Methods, Instruments, & Computers
2003, 35 (3), 364-368

Remote Web usability testing

ANDRES BARAVALLE and VITAVESKA LANFRANCHI
University of Turin, Turin, Italy

Recently, various techniques for and approaches to extending usability testing beyond the traditional laboratories and technologies have emerged. Remote usability testing allows researchers to evaluate the usability of websites by gathering information from remote users. Several different approaches have been proposed, but they often require that the user perform particular installations or configurations. We introduce OpenWebSurvey, a software system for remote usability testing that can remotely record users' behavior while they surf the Internet and that requires no program installation or configuration.

Many studies of behavior require technical means of monitoring responding or are facilitated by the use of technical tools.

The technical resources needed to perform Web usability tests are often inserted into the context of a usability laboratory. Less often, these technologies are taken directly *in loco* (e.g., into the user's workplace or home).

Performing a test in a usability laboratory requires a number of users to come individually to the laboratory and execute certain tasks under the supervision of an administrator and/or be recorded by special technologies. The main advantage of this approach is that it is possible to better control the circumstances that can occur during the test; moreover, it is less time-consuming and more convenient for the researcher to have all the participants in the laboratory.

This approach fails in situations in which it is difficult to find a sufficient (or significant) number of users willing and able to participate in the test. In fact, there are some categories of users who cannot or will not go to a usability laboratory, such as busy, ill, or disabled persons. Moreover, in some situations the researcher may not want the participating users to be in an artificial setting.

Conducting usability tests *in loco* makes it possible to test a user in a real-word environment, outside of artificial settings.

The drawback of this approach is that it is usually time-consuming and expensive, since the researcher and the equipment need to move from place to place for every test. Furthermore, there may be places that a researcher cannot easily access, such as hospital units or educational centers.

A controversial issue regarding both approaches is the adequate number of participating users needed to perform a significant usability test. Whereas some researchers have suggested that a small number of users is enough (Nielsen, 2000; Nielsen & Landauer, 1993; Virzi, 1992), others assert that important details can be overlooked when small numbers are used (Perfetti & Landesman, n.d.; Spool & Schroeder, 2001). According to the latter studies, a larger sample of users with different experiences and backgrounds could reveal a greater number of usability issues.

Remote usability testing allows researchers to evaluate the usability of websites by gathering information from remote users. This approach does not differ much from the *in loco* strategy: Usability evaluation is conducted in a real-world environment, but without the need to displace researchers and technologies to the users' sites.

The key advantage of remote testing usability is that a large number of users can take part with little incremental cost per participant, making it possible to overcome the trade-off between number of participants and test cost suggested by Nielsen (n.d.).

The efficacy of remote usability testing is still under debate, and some researchers (e.g., Ebling & John, 2000; Tamler, 2001) have pointed out a loss of information due to lack of interaction between the researcher and the user as well as to lack of direct observation.

Real-time observation may prove to be superior in some cases, since remote usability tests cannot provide real-time, subjective data about the user's experience. According to Tamler (2001), quantitative data are useful in uncovering the presence of a problem, but are often inadequate, since they may reveal the presence and magnitude of the problem but not its nature. Obviously, traditional laboratory tests allow the use of complex and sophisticated technologies to record user vocalizations and facial expressions and allow a test administrator to be present to guide the users, which Ebling and John (2000) suggested is critical.

By contrast, a comparison of laboratory-based and remote testing protocols (Tullis, Fleischman, McNulty, Cianchette, & Bergel, 2002) has suggested that, with limited instrumentation, both methods revealed the same core problems and resulted in similar task time and success measures. The goal of that study was to evaluate the effectiveness of remote usability testing by conducting

Correspondence concerning this article should be addressed to A. Baravalle or V. Lanfranchi, Department of Computer Science, University of Turin, C.so Svizzera, 185, Turin 10149, Italy (e-mail: andres@di.unito.it; vita@di.unito.it).

both laboratory and remote usability tests of two prototype websites and comparing the results. When task completion rates and task times were considered, the behaviors of the participating users were similar. Whereas laboratory testing proved essential to identifying certain usability issues and user motivations, remote testing was more advantageous for understanding problems unique to specific types of users and environments, since the number and, therefore, the diversity of the users were increased.

Regarding remote usability testing, three main approaches can be found in the literature. We will refer to them as the *Web browser*, *website*, and *proxy* approaches.

The Web browser approach is based on the development of a specific client-side application that the user will use to surf the Web. This application is usually a customization of a specific browser, usually Microsoft Internet Explorer or Mozilla. This kind of approach is adopted by Usable Tools (2001), among others.

The main disadvantages are that users are forced to install and use a particular client as well as a hardware and/or software platform, and that they will have to install the client themselves, which it is not always a quick and easy task.

The website approach is based on a Web application that is inserted into the website in question and monitors user behavior. It usually requires full access to the website code, so it can be used only in rare cases, since most websites do not allow external researchers to modify their sources. This kind of approach is adopted by Paternò, Paganelli, and Santoro (2001) , among others.

The proxy approach is based on an application that lies between the client and the server monitoring the navigation. The main advantage of this approach is that the tested users need not install anything on their computers, and that the researcher does not need to have access to the website code.

Our research aims to help resolve these issues by developing a proxy-based software system that is able to record, store, share, and process data for Web usability analysis.

OpenWebSurvey shares the proxy approach of WebQuilt (Hong, Heer, Waterson, & Landay, 2001), but the two proxy implementations are based on different technologies and have different aims. Most importantly, OpenWebSurvey gathers data about the user's environment and system characteristics (e.g., browser name, browser version, operating system, language), allows user identification through a login process, and also aims to be able to collect client-side data (e.g., mouse movements, mouse clicks) in the future. OpenWebSurvey allows researchers to use its server installation, so that even unskilled users can participate in a usability test without having to install the proxy onto a dedicated server.

An alpha release of OpenWebSurvey is being tested at the Department of Computer Science of the University of Turin. It is freely available under the terms of Gnu Public License or, on request, it can be used directly on OpenWebSurvey's main server (http://openwebsurvey.di.unito.it).

Method

Testing interface. As was already stated in the introduction, OpenWebSurvey is a Web-based software system for remote Web usability testing. When a user agrees to participate in a survey, she simply connects to the OpenWebSurvey website (using a Web browser), where she finds the necessary instructions. Generally, she has to surf one or more sites and answer some questions.

When the survey session starts, a two-frame Web page (Figure 1) is loaded. The upper frame contains the survey questions and input fields for the answers, whereas the lower frame presents a view of the website in question.

For each survey question, it is possible to have different layout views: Text fields are used to collect user answers, and radio buttons and pop-up menus are used to propose possible choices from which the user must select. If a question requires only the performance of actions and no explicit answer from the user, it is not necessary to show an input field, but only a button that allows the user to proceed.

The lower frame shows the site in question, modified to record surfing data. The navigation is left unmodified: The user can follow links, play sounds, or watch movies as usual.

Technical issues will be discussed in the following sections.

Survey questions. Survey questions are stored in triple-s eXtensible Markup Language (XML) format (Hughes, Jenkins, & Wright, 1999), an XML dialect[1] that was created to describe traditional survey questions. OpenWebSurvey uses an extended version of the language that is validated with standard triple-s document-type definition (DTD) and describes more complex visualization information. For example, for every question, a website to be loaded in the lower frame can be indicated.

Triple-s XML is transformed (on the server side) with eXtensible Stylesheet Language Transformation (XSLT) into eXtensible HyperText Markup Language (XHTML), so the user simply sees a Web page with questions on it.

Collected data. Survey answers and collected data are stored as serialized objects in a database (OpenWebSurvey uses a MySQL database through an abstraction layer).[2]

In addition, OpenWebSurvey offers functions of importing and exporting the collected data in XML or CSV formats. These are both standard formats and can be imported into most statistical software, such as SPSS.

Analysis stage. OpenWebSurvey does not aim to provide analysis mechanisms but only to gather and store data about user behavior and answers that can be used to perform in-depth analyses.

As was stated above, statistical operations with the collected data can be performed by importing data into specific programs.

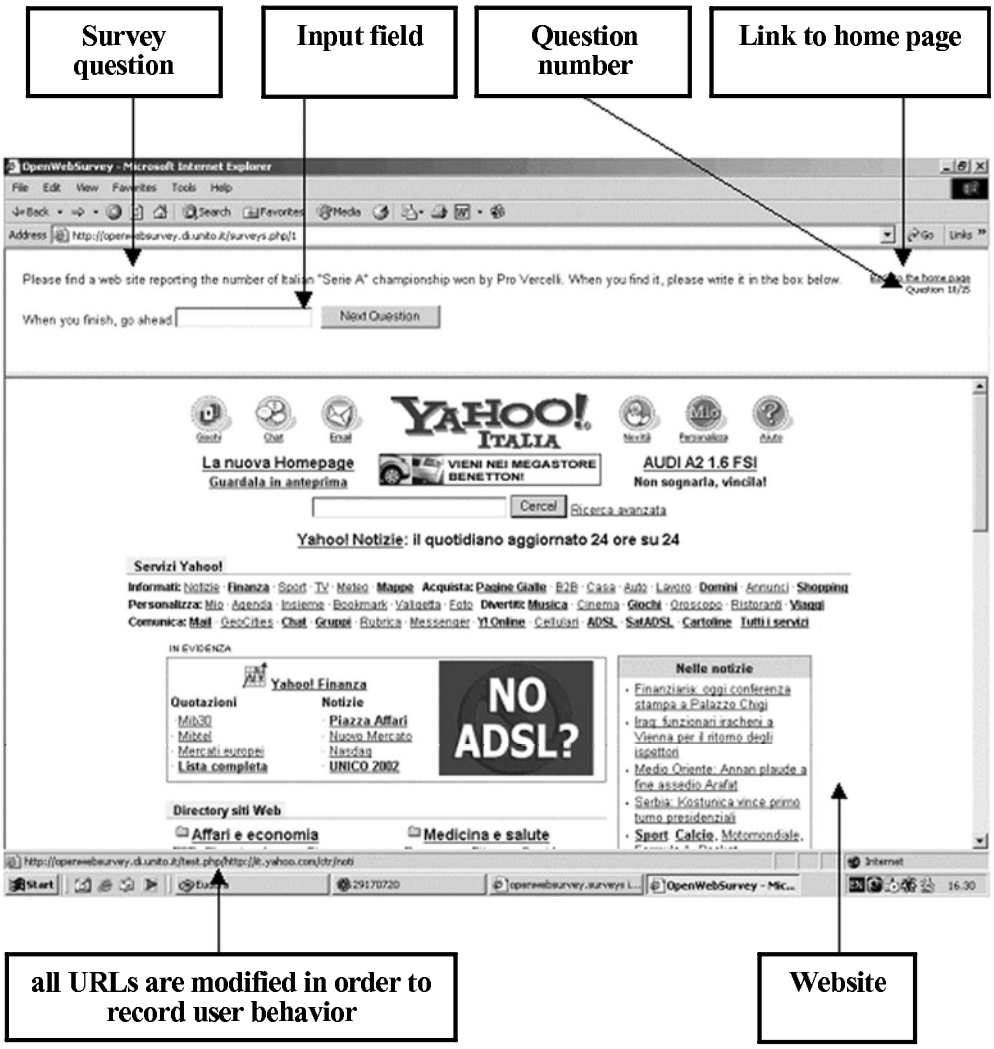

Figure 1. OpenWebSurvey interface.

The usefulness of OpenWebSurvey is maximized if it is used with survey questions that involve the performance of tasks such as looking for a text on a website. In this way, it is possible to analyze not only the answer that the user found, but also how she found it.

It is possible to understand specifically how much time a user needed to find an answer and how she found it. For example, in a study of strategies in using search engines such as Yahoo, it might be useful to know if the user performed the requested task using a free search or a category search. In the first case, it might be useful to know which key words she used and how long it took her to succeed in executing the requested task.

System architecture. A user surfing the internet uses a browser, which is responsible for sending page requests to the Web server.

OpenWebSurvey acts as a proxy server between the browser and the Web server. It retrieves requested Web pages, modifies them, and then sends them back to the client (see Figure 2).

At the beginning of the test, OpenWebSurvey assigns the user a unique identification code and saves general information about the user's environment, such as operating system, browser name and version, and language; such data can be useful in verifying if a usability issue may be related to particular system configurations.

The second step is to connect to the starting URL and retrieve the page. Links on the page are extracted and rewritten with the use of Perl-Compatible Regular Expressions.[3] The final step is to serve the requested URL with the modified links that point to the OpenWebSurvey server, to use the proxy again on the next page request.

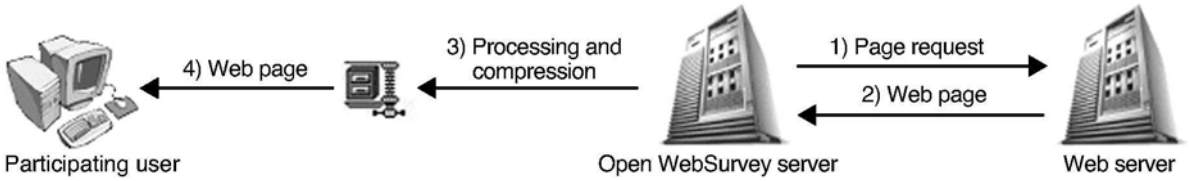

Figure 2. OpenWebSurvey architecture.

If the link is http://www.mysite.com, then it will be rewritten as: http://openwebsurvey.di.unito.it/test.php/http://www.mysite.com.

The link http://www.mysite.com is sent as a parameter to the OpenWebSurvey server, introducing a recursive mechanism. Any link the user follows will be served through the OpenWebSurvey server. Thereafter, all links will automatically be rewritten so that the user can continue using the proxy.

The two frame windows act separately, each collecting and storing different data: The upper frame saves data about user answers to survey questions, and the lower frame saves data about pages visited during the test, load time of each page, total visit time, page visit time, and user input data (as text inserted on forms).

In order to provide effective performance, OpenWebSurvey uses a compression algorithm[4] to serve the pages to the client and uses caching to serve the survey questions. OpenWebSurvey is written using PHP Hypertext Preguages, and database tables and XML for data storage.

Capabilities. OpenWebSurvey is able to process Web pages written in valid HTML or valid XHTML—that is, pages that are verified against the associated DTD. In most cases, nonvalid HTML pages can also be processed, since the use of regular expressions makes the system quite tolerant to this type of errors. It can also process JavaScript code if it finds full URLs (e.g., http://www.mydomain.com/mypage.html).

OpenWebSurvey failures may be due to (1) human error—a user can choose to leave the test, use the address bar to browse, or close the browser; (2) external problems—OpenWebSurvey will fail if there is a failure in the network connection, Web server, or database server; or (3) system limitations—OpenWebSurvey is not able to process links in binary data, such as objects managed by plug-ins (e.g., Flash or Director movies, Java applets) or JavaScript URLs built with concatenation techniques. OpenWebSurvey is only partially able to handle redirections or security checks based on HTTP headers.

If OpenWebSurvey fails due to user or external errors, the test fails and the user must restart it. If it fails because of system limitations, it simply does not record the user's behavior, but the user can continue browsing.

The user's operating system and browser do not affect OpenWebSurvey's performance.

Discussion

OpenWebSurvey makes it possible to perform usability tests remotely, gathering quantitative data about the user's surfing behavior.

For psychological analysis, it is quite useful to have a large sample of users to be tested, so that statistical analyses regarding attitudes and behaviors can be performed. Through OpenWebSurvey, a test can be performed on a significant number of users: Recording behavior remotely is not expensive, since it requires no special equipment. It is thus possible to overcome the cost/benefit trade-off pointed out by Nielsen (n.d.).

REFERENCES

Ebling, M. R., & John, B. E. (2000, August). On the contributions of different empirical data in usability testing. In *Proceedings on Designing Interactive Systems Conference* (pp. 289-296). New York: ACM Press.

Hong, J. I., Heer, J., Waterson, S., & Landay, J. A. (2001). WebQuilt: A proxy-based approach to remote Web usability testing. *Association for Computing Machinery Transactions on Information Systems*, **19**, 263-285.

Hughes, K., Jenkins, S., & Wright, G. (1999). *Triple-s XML: A standard within a standard.* Retrieved August 1, 2002 from http://www.triple-s.org/sssasc99.htm.

Nielsen, J. (2000). *Why you only need to test with 5 users*. Retrieved May 20, 2003 from www.useit.com/alertbox/20000319.html.

Nielsen, J. (n.d.). *How to conduct a heuristic evaluation.* Retrieved August 1, 2002 from http://www.useit.com/papers/heuristic/heuristic_evaluation.html.

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In S. Ashlund et al. (Eds.), *Proceedings of the Association for Computing Machinery Interchi'93 Conference* (pp. 206-213). New York: ACM Press.

Paternò, F., Paganelli, L., & Santoro, C. (2001). Models, tools and transformations for design and evaluation of interactive applications. In *Proceedings of the Human Computer Interaction Conference* (pp. 23-28). Patras, Greece: Typorama.

Perfetti, C., & Landesman, L. (n.d.). *Eight is not enough.* Retrieved May 20, 2003 from http://www.uie.com/Articles/eight_is_not_enough.htm.

Spool, J., & Schroeder, W. (2001). Testing websites: Five users is nowhere near enough. In *Proceedings of the Association for Computing Machinery, Computer Human Interaction Conference* (pp. 285-286). New York: ACM Press.

Tamler, H. M. (2001). High-tech vs. high-touch: Some limits of automation in diagnostic usability testing. In HT Consulting, *Sample usability evaluation report* (on line). Retrieved May 20, 2003 from http://www.htamler.com/papers/techtouch.

Tullis, T., Fleischman, S., McNulty, M., Cianchette, C., & Bergel, M. (2002). An empirical comparison of lab and remote usability testing of Web sites. In *Usability Professionals' Association*

2002 Conference Proceedings (CD-ROM). Bloomingdale, IL: Usability Professionals' Association.

USABLE TOOLS (2001). *Usability browser.* Retrieved August 1, 2002 from: http://www.usabletools.com/ub.htm.

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

NOTES

1. Languages based on XML, usually called *dialects*, use custom tags that convey the semantic value of the described data.

2. The OpenWebSurvey database abstraction layer is based on PHP Base Library, commonly known as phplib. Thus, it is possible to use any database supported by PHP, according to the preferences of the administrators of the site using the OpenWebSurvey software system, by developing the necessary interaction classes.

3. Perl-Compatible Regular Expressions (PCRE) is a syntax written by Philip Hazel that implements pattern matching. PCRE is used by different projects, including Python, Apache, and PHP.

4. OpenWebSurvey uses GNU zip (gzip), a variation of the Lempel-Ziv algorithm, to serve compressed pages. Most browsers (e.g., current versions of Mozilla, Netscape, and Explorer) support pages compressed using gzip, shortening the download time of Web pages.

(Manuscript received October 1, 2002;
revision accepted for publication June 2, 2003.)