Moving Usability Testing Onto the Web

Martin Svensson, Arnold Johansson, Anna-Lena Ereback, Kristina Höök,
Jussi Karlgren
SICS, Box 1263, S-164 28 Kista, Sweden.
+46 8 752 15 00 {martins, arnie, annalena, kia, jussi}@sics.se
Ivan Bretan
Telia Research AB Vitsandsgatan 9, S-123 86 Farsta, Sweden.
+46-8-713 51 40 Ivan.P.Bretan@telia.se

Abstract: In order to remotely obtain detailed usability data by tracking user behaviors within a given web site, a server-based usability testing environment has been created. Web pages are annotated in such a way that arbitrary user actions (such as mouse over link or click back button) can be selected for logging. In addition, the system allows the experiment designer to interleave interactive questions into the usability evaluation, which for instance could be triggered by a particular sequence of actions. The system works in conjunction with clustering and visualization algorithms that can be applied to the resulting log file data. A first version of the system has been used successfully to carry out a web usability evaluation.

Introduction

Usability testing is a difficult and costly process. One problem is to find subjects willing to take part in the experiments usability engineers set up. Another is to actually perform the testing, an often time-consuming and costly process. Once the data is collected, yet another problem is to analyze the collected data on user performance and draw conclusions from the result. These problems are far from the only ones involved in usability testing, but they constitute major drawbacks that make most software developers reluctant to actually perform them.

In a project named Traces, we have tried to tackle these problems through building a usability laboratory that will work remotely over the Internet. We want to test the usability of web pages, and, in an extension we see the possibility of testing other interfaces built using Java or other tools that allows for more interactive interfaces.

Our approach has been to support the usability engineer in several steps of the test cycle. First, early on in the design cycle, the usability engineer might want to try out ideas using a 'Cognitive Walkthrough' or 'Heuristic Evaluation' method [Lewis C et al. 1990; Nielsen and Mack 1994]. This can be done through putting some mock-up pictures on the web and then have experts evaluate the design through using our tool. Later in the design, before putting the system out on the market, end users can be involve in a controlled study [Holyer 1993]. The tool then needs to log users interacting with the system, perhaps posing some questions to them before, after or even during use. Finally, once the system is in use, user feedback can be obtained on how to improve the system through questionnaires.

Our idea is simple and straightforward: to build a logging tool that can access all the 'events' that takes place in a web browser and that can interact with the user at predefined junctures in order to get explicit feedback. When we turn to the web, we gain access to many users, thus we avoid some of the problems involved in finding users that are willing to spend time on testing software. Also, users can perform the test at any time or place they find convenient. This reduces the need for laboratory testing thus tackling the second problem mentioned above.

Unfortunately, logging events over the web will not avoid the problem with long logs that the usability engineer has to analyze. Depending on what you decide to log, you might end up with very much data. Our approach to solving this is to look at various methods, such as statistical, demographics, or clustering methods, and combine them with visualization tools such as Spotfire [Ahlberg and Shneiderman 1994a; Ahlberg and Shneiderman 1994b] and HyperSpace (http://www.cs.bham.ac.uk/~anp/haiku).

In [fig. 1] we can see the kind of scenario we envision: first the usability engineer sets up the design to be evaluated, then the Traces logging tool collects data on how end users or experts go through the interface (commenting or just logging or both), and finally we present the obtained data using various tools and methods.

Depending on how advanced the interface to be tested is, we can catch various different usability problems. Examples of usability problems we can catch are:

- structural, navigational problems,
- design of graphics, icons, text links, etc.,
- anticipations, attitudes, and appeal of the interface.

So to summarize our tool has to:

- catch user events
- enable the usability engineer to decide which events to catch for a particular study
- keep track of the time when an event occurred
- be able to receive comments from the user/expert and be paused
- preserve the original interface of that being tested

Let us start by describing our implementation, and our ideas for visualization. We then describe an experimental study performed using our tool. Finally, we summarize and discuss future requirements on this kind of usability testing environment.

Implementation of Traces

Starting from the requirements outlined in the previous section, the Traces environment was developed. The Traces environment is a two-component tool implemented in Java and JavaScript. The first component allows designers to annotate web pages and the second logs users' actions. That is, when a user enters a site, prepared for Traces, our tool logs every event that occurs inside the web pages in that site. Let us start by describing the logging tool.

Communication between Client, Traces, and Server

HTML the standard for web-content, is constantly changing. The different browsers offer all kinds of functionality, such as event handling, security restrictions, etc. Although Traces is designed for Netscape the intention is to present a solution that can easily be extended to other browsers and conditions. Thus, it is important to make Traces as general as possible. Traces must:

- make no assumption on what browser it is supposed to work with,
- make no access to the HTML-pages it is going to log, in effect, accepting any version of the HTML language and all browsers that support it.

This means that Traces can not derive any information from the HTML-pages, nor can it rely on any browser specific information. That is, Traces has to be an autonomous program that the annotated web pages can send messages to. To achieve this, applets, JavaScript and frames were used. The annotated web pages contain JavaScript that send messages to an applet, Traces, that in turn time-stamps the messages and sends them back to the server. The browser window is divided into two parts – using frames – one that contains the applet and one that contains the annotated web page. This way the applet is kept in the browser rather than loaded with each new page the user enters.

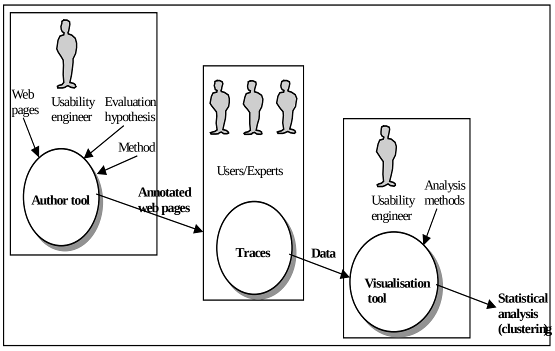

Figure 1: Usability interface testing with Traces.

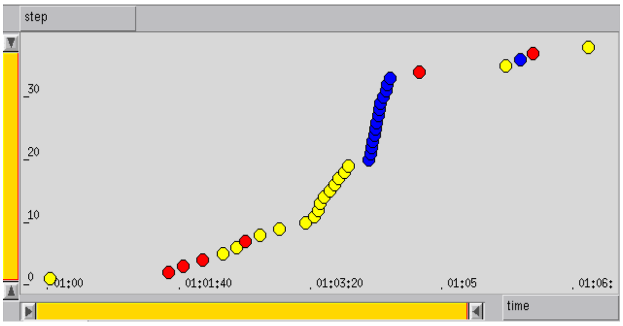

Figure 2: Spotfire visualising step-link name for one user

Using two frames solves another problem. We need to introduce some control and comment fields to enable users to start and stop the logging, make comments, etc. We decided to add this information to the frame containing the applet, since it needs to be visual all the time. However, this information can be removed to fit with the usability testing situation.

Event Handling

It is necessary to catch user events and store these in a readable fashion. We have to be careful in selecting user events; the amount of collected data can very fast grow out of proportion. Recording low-level user events, such as keystrokes, produces vast amounts of data, and the method is thus usually discarded when evaluating user interfaces. As stated earlier Traces must allow the usability engineer to decide which events to log.

Most browsers have an event handling mechanism that executes JavaScript code. Different events trigger different parts of the JavaScript code. For instance, when a user clicks on a link a handler could be triggered and send the message link hallo to Traces. Event handlers are easy to implement and Traces does not need any knowledge about the browser. All communication between Traces and the HTML-pages is done through JavaScript code added to the HTML-pages that sends messages to Traces.

Authoring environment

Now let us turn to the second component of our tool, the authoring environment. Traces does not retrieve any information from the HTML-pages or the browser. All events are sent through JavaScript from the HTML-pages to Traces. This limited communication means that the detailed information about an event must be included in the messages sent from the HTML-pages. So in order to annotate a web page, the usability engineer either has to be skilled in how to produce JavaScript code that sends back the right kind of information to Traces, or we must build an authoring tool that supports the editing and making of new annotated web pages. The latter is to be preferred, but places the following demands on the authoring environment:

1. It must assist the usability engineer to create annotated web pages. Depending upon method used and stage reached in the design life cycle, different kinds of logging will be needed. In a site that contains hundreds of HTML-pages it is crucial to be able to quickly change the HTML-pages for different kinds of studies.
2. It should help the usability engineer to set up questionnaires and pop-up questions.
3. It should preferably have knowledge enough to be able to recommend to the usability engineering what an appropriate message to Traces could be. An HTML-page contains many properties that can be accessed through JavaScript, out of which only some are interesting in a particular situation.
4. It should support general guidelines on how to set up a study using the different usability testing methods available.

So far we have only implemented an authoring tool that addresses the first and second issue. The authoring tool allows the usability engineer to specify what events are to be caught and what messages are to be sent to Traces. Furthermore the designer can create pop-up questions or surveys that will appear in specific web pages. The authoring tool then parses one HTML-file or a directory structure and produce new HTML-file(s) that supports Traces, i.e., the new HTML-files contains JavaScript event handlers that call the Traces applet.

Related Work

The idea of using WWW as an environment for usability testing is not new. Catledge and Pitkow [Catledge and Pitkow 1995] captured client-side user events through creating a new version of Xmosaic. Their tool differs from ours in two important ways. First, they had to implement a specific browser. Second, their tool was passive, in the sense that it did not allow for any interaction with the user apart from start-up question that informed the user about the logging. Even if their technical solution was not optimal from our perspective, they did show that it is possible to get useful data from capturing user events, both in terms of identifying browsing strategies, and making design improvements.

Chang [Chang and Dillon 1997] use a slightly different approach when they move the usability laboratory to the users. Their program work as a component in the user's machine, allowing them to catch every event in the system. Thus, Chang and Tharam are not restricted to the interface that is to be tested, but they can also monitor what is going on outside the interface.

In addition there are a large number of tools that collect statistics, such as, how many hits a web page has, what machine accessed the page, and what pages the user visited before she accessed the current page [Pitkow 1997]. These tools are frequently used for marketing purposes but in some cases also used for usability purposes [Brown and Benford 1996] and WebVIP (http://zing.ncsl.nist.gov/~webmet/vip/webvip-process.html) or other purposes such as Footprints (http://wex.www.media.mit.edu/people/wex/Footprints/footprints1.html). We believe that there is a need for tools that go beyond the logging of "accessed" web pages. It is also important to catch what is going on within a web page, and also to have the ability to interact with users by, for instance, pose pop-up questions to them. So basically what makes Traces unique:

- the ability to create continuous logs of each individual user's session in a site,
- the possibility to log all or just some events that a browser allows for,
- the possibility to create interactive questionnaires and pop-up queries integrated within the web pages, where the answers from the users are integrated with the logs obtained.

Visualization and Clustering

As mentioned previously, Traces can provide us with long and detailed logs of user behavior. These need to be analyzed when we establish usability of a site. Since our approach is to provide a fairly general tool for usability testing, it will not be possible to find one single tool that can perform all sorts of statistics needed. Instead we need to find combinations of visualization of data, statistical analysis, clustering, etc. that each contribute to the analysis.

In our first study, we have devoted attention to the following usability areas relevant mostly to web site design:

- navigational tools
- browsing flow and space
- the connection between demographic data and user behavior

Below we discuss the study and how each area can be dealt with using available tools and methods. We do not want to claim that we have found general usability guidelines for the design of web pages – rather this exercise is meant to convince us that Traces can provide the usability engineer with relevant information and that this information can be analyzed through a combination of tools available on the market.

Test study

Our study was performed on a subpart of the site Passagen(http://www.passagen.se). The subjects were students in computer science and cognitive science. They were asked to comment on their actions by typing remarks in the logger's visual interface. They were also asked to provide additional demographic data, such as gender, age, level of browsing knowledge, and address. The subjects then performed three predefined tasks:

- To find a particular link somewhere in the site, click on it, and then return to the main page.
- To find a link that leads to an article, and then browse around this article (consisting of several pages) until they were satisfied.
- To follow detailed instructions leading them step by step through a number of pages.

These three tasks reflect three different situations: a search situation, a browsing situation and one control case allowing us to verify the logging tool.

We received about 60 log-files, out of which, in the end, only 33 were used, since the tasks had been fully completed in them. We made the decision to not include the incomplete logs in our statistical analysis mainly due to the fact that we could only guess the reasons for the participants to not complete the tasks given to them. However, this large dropout rate tells us something about the logger when doing these kinds of studies. The tool has to be totally reliable in order to rule out that the cause for the uncompleted logs is not due to any malfunction in the logger. 21 male subjects and 12 of female subjects participated. Among the men, 9 considered themselves to be expert web users, 7 good and 5 intermediate users. Among the females only 2 considered themselves to be experts, 4 good and 6 intermediate. The fastest subject completed a logging in 16 steps (1.27 minutes) and the slowest in 38 steps (12.32 minutes). The average value was 24 steps (5.41 minutes) and the median value 24,2 steps (5.52 minutes). The oldest person in the study was 44 years old and the youngest 21.

Site Structure Analysis

Depending upon content, task, targeted user group, etc., a site's structure can be organised differently to be most efficient and usable. In general, a site should be well balanced and have its information *structured* in some kind of intuitive *hierarchy*, e.g. a balanced tree structure were different branches in the tree contain different topics [Nielsen 1993; Shneiderman 1992; Shneiderman 1997]. It is also important that the site has a good level of *connectivity*, i.e. number of links between pages. Again, the correct grade of connectivity can not be determined through any general standards, but must be decided from site to site and from page to page depending on the information content, users, etc. Index pages may contain numerous links, while pages with text documents may not have any links at all.

We used Hyperspace to examine a site's structure and connectivity. This tool is a dynamic, self-organising interface for real-time data mining. As a user browses through web pages in the browser, Hyperspace creates a 3D molecular like structure of the pages. In this structure spheres represent pages, and lines between the spheres represent links on a page leading to another page. The structure can be turned in any angle and the user can zoom in and out.

The model generated through interacting with Hyperspace was then examined in terms of structure and connectivity. Both the site as a whole and subparts of it were examined in this way. From these models we found that the site had a flat tree structure which would indicate that it is easy for the users to orient themselves, but on the other hand, there were some confusing links that lead several steps down in the hierarchy. These links were no doubt meant to be highlighted shortcuts (which could be a good idea) but they tended to be rather confusing in this context.

The connectivity of the site was in general good, apart from one subpart lacking connections with any of the main pages. This means that it will be difficult for users to jump back to the main page of the site, thus leading to orientation problems.

Browsing flow and space

It is useful to examine users' browsing behaviour in order to find design mistakes. For example, it is valuable to know if users are aware of various possible links from a page, e.g. buttons, text, menus etc. If not, we might need to redesign such icons or text links or the general site structure.

In order to understand our targeted user group's browsing behavior, we followed and analysed the subjects' actions by stepping through the log-files for the first (search) task in our study (see above). First we followed each user's path with Hyperspace to create a model over the site that the user had visited. This model was later compared to the complete structure of the site and models from other subjects performing the same task. By comparing these different models we found that about 60% of the subjects moved in almost exactly the same space in the site and almost all of these found the direct link to the target page that was located on the main page.

With use of Spotfire we then analysed each subjects' path through the test web site. Spotfire displays data in a multi-dimensional graph space that can be manipulated interactively.

By selecting time on the x-axis and link steps on the y-axis, see [fig. 2], we got a clear picture of how much time was spent on each page.

From analyzing the whole group of subjects, we found that most users had a slow start but then moved fairly quickly from page to page with an average time on each page of 5-10 seconds. As with Hyperspace we could see that most of the users found the correct page and that they kept to almost the same set of pages. We also discovered that the different users needed very varied amounts of time to complete their tasks. Subjects spent from less than two minutes to more than twelve minutes to complete the first task. Completion time was later checked against the subjects' age, gender, and experience but no definite conclusions could be made due to the small amount of subjects.

Orientation and link types

The next issue to examine is the subject's understanding of the available navigational aids, such as menus, icons and textual links. We examined how subjects navigated in the test site and which kinds of aids they used. On many of the pages in the site – especially those in the top layers – there exists homogenous, graphical navigational bars with links that can take a user to the main page, to the main sub-pages, to a search page and to a contact page. On some pages the same links are displayed both as text and graphics.

For example, for the particular subjects’ behavior displayed in [fig. 2], we can see a pattern where several dark (Spotfire displayed as blue dots) follow quickly after one another. These are in fact back -button events, and shows how the user has read the article and is now trying to return to the main page. Since there was no direct link back to the main page, the back -button had to be used: an example of a potentially bad design. This user, and most other in our study relied more on the back-button than on the orientation links on the pages to get back to the main page. Quite a few subjects first used the back button until they arrived at a page with an orientation link that could take them to the main page; no-one solely used links. This could depend on various things such as a sense of more familiarity and trust in the browser’s interface or a bad design or location of the pages orientation interface.

An analysis to differentiate user preferences for text or icon links revealed an interesting pattern which reflected page and site design. We performed an X^2-analysis in which we looked at series of link usage. We distinguish between the following user actions: a) user follows a text link if only text links are available - T - or if there is a choice between text and graphics - T(G); b) user follows a graphical link with only graphics available - G - or when there is a choice between text and graphics G(T); c) user uses back button B.

In [table 1] the data are tabulated by adjacent link choices: the x-axis represents the user following a link, and the y-axis the *NEXT* link followed. Thus, for example, we see from the table that there are two occurrences of first choosing a graphical link on a page that has both text and graphical links, and then choosing a text link an a page that also has both has text and graphical links (i.e. the 2 in the third column and second row of data).

	T	T(G)	G(T)	G	B
T	5	2	0	0	0
T(G)	2	3	2	0	0
G(T)	1	1	1	1	0
G	0	1	0	1	0
B	0	1	1	0	13

Table 1: The X^2 matrix for user X.

The X^2-analysis (on the entire matrices, or on subsets of them) revealed with a 97% probability that most subjects did not choose to follow only text links or graphical links. Instead they switched between the two repeatedly by for example first taking a text link, then a graphical link, then a text link again etc. This result seemed somewhat surprising at first but could be explained by the structure of the site and the design of the pages in the site. The pages contained different amounts of text, which sometimes meant that the navigational bar was placed so far down in the page that the subjects had to scroll to it. To further investigate why users choose a specific link type over another on a certain page, we looked at the main page which has several double-links, i.e. two differently looking links – one text and one graphic – leading to the same location. The results here were about the same as in the X^2-analysis – the subjects seemed to be using both of the link types.

Connecting demographic data to user behavior

If demographic information about the subjects can be obtained, it can possible be linked to the behavior of groups of subjects. This in turn means that we can understand why certain user groups behave differently and possibly create better designs to fit them

Our demographic data (including the subjects’ gender, age and knowledge of browsing) provided us with a good base for clustering different groups of users with respect to their behavior. We examined, for example, how fast the different groups of users completed the tasks and if they used any special browsing technique. Our group of subjects was small and homogenous, and did not give us purchase to infer statistically significant differences between categories of users, but this area shows promise for further investigation.

Usability issues not covered

Numerous other usability issues could potentially have been included in our study. For example, see [Burger and Jackson 1997], list a number of usability problems related to the appeal of a web site. These kinds of ‘soft’ issues will not be captured by logging user behavior alone, but in combination with asking the user (through using the interactive querying mechanisms of Traces) what they expect or how they react to various aspects of the interface, some of these ‘soft’ issues can be caught.

Summary/Conclusions

Usability testing must be altered to fit with the rapid pace by which applications are developed (sometimes collapsing the whole software development cycle into one step). The proposed Traces environment enables us to perform studies rapidly with large amount of distributed users in their regular work environment. The advantage of Traces lies in its ability to produce continuous logs of user events and possibilities to interact with the user in various ways during a test-session. Usability issues addressed are navigation, design, and subjective feedback.

What have not been discussed here are the privacy issues: when we make it possible to log each user in detail, we can also imagine various not so user-friendly scenarios. Our tool might very well be abused to check on how well users perform their work tasks, direct advertising to them that they have not asked for, etc. While we recognize these problems, we still find the strengths of this approach to usability testing to outweigh the potential dangers of misuse of the tool.

References

[Ahlberg and Shneiderman 1994a] Ahlberg, T. and Shneiderman, B. (1994). AlphaSlider: A Compact and Rapid Selector. *Proc. ACM Conference on Human Factors in Computing Systems '94*, ACM, 365-371.

[Ahlberg and Shneiderman 1994b] Ahlberg, T. and Shneiderman, B. (1994). Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proc. ACM Conference on Human Factors in Computing Systems '94*, ACM, 313-317.

[Brown and Benford 1996] Brown, C. and Benford, S. (1996). Tracking WWW Users: Experience From the Design of HyperVisVR. *Proc. WebNet '96*, AACE, San Francisco, CA, 57-63.

[Burger and Jackson 1997] Burger, K. Jackson, E. (1997). Usability Evaluation Techniques for Large-Scale Web Sites, *Proc. Human-Computer Interaction INTERACT ´97*, IFIP, Sydney, AU, 571-572.

[Catledge and Pitkow 1995] Catledge, L D. and Pitkow, J E. (1995). *Characterizing Browsing Strategies in the World Wide Web. Computer Networks and ISDN Systems*, 27, 1065-1073.

[Chang and Dillon, 1997] Chang, E. and Dillon, S.T. (1997) Automated Usability Testing, *Proc. Human-Computer Interaction INTERACT '97*, IFIP, Sydney, AU, 77-84.

[Holyer 1993] Holyer, A. (1993). Methods for Evaluating User Interfaces. *Research Paper No. 301*, School of Cognitive and Computing Sciences, University of Sussex, UK.

[Lewis C et al. 1990] Lewis, C., Polson, P., Wharton, C., Rieman, J. (1990). Testing a Walkthrough Methodology For Theory Based Design of Walk-up and Use Interfaces. *Proc. ACM Conference on Human Factors in Computing Systems '90*, ACM, New York, NY, 235-241.

[Nielsen 1993] Nielsen, J. (1993). Iterative user interface design, *IEEE Computer*, 26, 32-41.

[Nielsen and Mack 1994] Nielsen J, Mack R L. (Eds.) (1994). *Usability Inspection Methods*, New York, NY: John Wiley & Sons Inc.

[Pitkow 1997] Pitkow, J. (1997). In search of Reliable Usage Data on the WWW. *Proc. Sixth International World Wide Web Conference*, Santa Clara, CA, 451-463.

[Shneiderman 1992] Shneiderman, B. (1992). *Designing the user interface – strategies for effective human-computer interaction*, 2nd Ed., Reading, MA: Addisson-Wesley.

[Shneiderman 1997] Shneiderman, Ben. (1997). Direct Manipulation for comprehensible, predictable and controllable user interfaces. *Proc. ACM International Workshop on Intelligent User Interfaces '97*, ACM, New York, NY, 33-39.

Acknowledgement

We would like to thank Nomos Management and Telia Research AB that together with NUTEK and SICS have co-financed this project. Thanks also to Preben Hansen and Fredrik Espinoza for valuable input. Finally we want to thank Lars Olsson and Paul Saretok for their continuous improvement on Traces.