A Method for Remote and Semi-Automatic Usability Evaluation of Web-based Applications Through Users Behavior Analysis

Ariel Vargas
Institute of Computing
University of Campinas
P.O. Box 6176, 13084-971,
Campinas, SP, Brazil
vargas@ic.unicamp.br

Harold Weffers
Laboratory for Quality Software
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB,
Eindhoven, The Netherlands
h.t.g.weffers@tue.nl

Heloísa Vieira da Rocha
Institute of Computing
University of Campinas
P.O. Box 6176, 13084-971,
Campinas, SP, Brazil
heloisa@ic.unicamp.br

ABSTRACT
In this paper we describe a method for evaluating the usability of web-based applications. Our method is based on remote and automatic capture and semi-automatic analysis of users behavior, in order to find usability problems in the applications' interfaces. The goal of our method is to allow an analysis of the way users actually interact with the evaluated interface. Through the analysis of users behavior is possible to find patterns of interaction. Analyzing the patterns found and comparing it to the expected behavior for the tasks performed by users, we can detect usability problems. In this paper we also briefly describe a first experiment with our method and some initial results that point to the potential of the method in performing remote and automatic usability evaluations.

Author Keywords
Usability evaluation, remote usability evaluation, user activity tracking, log file analysis, semi-automatic analysis, web usage mining, pattern mining, sequence mining, user behavior analysis, user experience.

ACM Classification Keywords
H.5.2 User Interfaces: Evaluation/methodology

General Terms
Ergonomics, Human Factors, User Issues, Graphical User Interfaces (GUI), Web-based interaction, Pattern analysis, Experimentation.

INTRODUCTION
In the last years the web has become a common environment for computer applications. This has aroused a growing interest about the usability of these applications among researchers and developers. In order to develop a good web-based application interface it is important to consider its usability since the beginning of the development process and throughout the lifetime of the application. Regardless of the development methodology applied, it is important to evaluate the application's usability during and also after the end of the development [6]. There are many methods to evaluate graphical user interfaces of computer applications. The most popular method is the user test, in which an evaluator observes the user behavior during his interaction with the interface, in order to detect usability problems [11]. Observing users behavior is an efficient method to find usability problems in the interface, however it is an expensive method due to the costs of finding users to test, moving them to the test laboratory, preparing the infra-structure, carrying out the test, collecting and analyzing the results [10,4,9]. Due to these costs it is common just to analyze the behavior of a few users in user tests. Furthermore, analyzing the behavior of a few users hinders a quantitative analysis giving to the evaluation just a qualitative feature. Some problems could just be highlighted in a quantitative analysis and also the impact of them could just be evaluated analyzing a large number of users [9,12]. Another important factor when the usability of an interface is evaluated is its context of usage, which is difficult to simulate in a user test made in a laboratory. To deal with the difficulties mentioned above we propose a method for usability evaluation based on remote and automatic capture and semi-automatic analysis of users behavior in order to find usability problems in web-based application's interfaces.

RELATED WORK
According to [2], in a remote and automatic usability evaluation users and evaluators are separated in space and time, i.e., users perform their interaction with the application in their usual work environment without moving to a test laboratory. Evaluators do not watch users during their interaction with the application. They just analyze their behavior afterwards. To perform this kind of analysis, the interaction of all users, performed in the application's

interface, is captured in a log file. Software tools analyze these captured data afterwards. There are some tools already developed to perform this kind of usability evaluation as: WebVip [16], WET [8], TEA [12] and UsaProxy [3], which just perform the capture of users data but do not analyze it. WebVip and TEA logs are similar to web server logs, with no detailed information about actions performed by users such as, mouse clicks and mouse movements. WET performs more detailed capture than WebVip. However, WebVip and WET need manual insertion of code in each web page to make the capture. TEA needs to be installed in the client's machine. UsaProxy, differently from the previous approaches, makes detailed capture of users actions and has the advantage of working as a proxy between server and client. It does not need any manual insertion of code in web pages or installations in client or server. Besides the capture tools mentioned, there are tools which perform the capture and some analysis of the data captured as: WebRemUsine [13], AWUSA [17], WAUTER [5], WELFIT [15] and WebQuilt [9]. WebRemUsine and WELFIT capture detailed information about users actions, however they need to insert some code in each web page on the server in order to make the capture. WebRemUsine analysis compares the expected sequence of actions for a task to the sequence of actions performed by a user. As a result, the tool shows the differences between both sequences. WELFIT performs an analysis of all users actions on a single page but does not care about the path followed between pages. AWUSA works with server logs and due to this it has no precision in capturing users actions. Its analysis is the same as WebRemUsine, comparing two sequences of actions and showing the differences. WAUTER captures users interaction in the client side. Its capturing is more precise, getting all actions performed by users in the web application, but it needs to install software on the client's computer. WAUTER makes its analysis just like AWUSA and WebRemUsine, comparing expected and performed sequence of actions for a task. WebQuilt works the same way as WAUTER capturing actions in the client side, its capture is however not so accurate, just getting information about pages requested to the web server. Nevertheless, its analysis is smarter than other tools referred before. It gets all actions performed by all users and groups them to find the most common navigation path for the pages of the application.

PROPOSED METHOD

Our proposed method for performing usability evaluation based on remote and automatic capture and semi-automatic analysis is called WebHint. It is composed of 3 steps as described below and shown in Figure 1.

Step 1 - Task Definition

The first step in the WebHint method is the definition of the tasks to be analyzed in the evaluation. A task is a sequence of actions performed by users in the application's interface with a specific goal. Ex: The sequence of actions performed to read an e-mail message, in a webmail application, constitutes a task. In Step 1, an evaluator defines the tasks performing the expected sequence of actions for each task in the interface of the evaluated application, as was planned by the application designer. The sequence of actions performed is captured by software and saved in a log file.

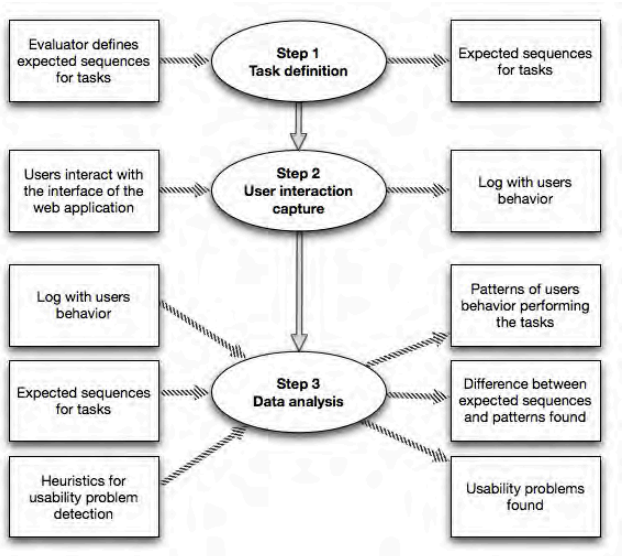

Figure 1. The WebHint proposed Method.

Step 2 – Users Interaction Capture

In the step 2 the users interaction with the application interface is monitored. All actions performed by all users in the web application's interface are captured: mouse movements, keystrokes, links accessed, pages loaded, etc.

Step 3 – Data Analysis

This is the most important step in the evaluation. Here all users actions captured in the step 2 are analyzed, as shown in Figure 2 and described below.

The first activity in the analysis step is the extraction of the tasks performed by users from the log files. For this purpose some algorithms were implemented. They are explained below: 1) Split log – the algorithm splits the log with all interaction of all users in several log files containing the interaction of one single user per file. 2) Determine sequence intervals – the algorithm finds in each log file the intervals in which there is a sequence of actions that represents the execution of a certain task. This is made looking for representative actions in the task as the "begin" and "end" points. The algorithm also deals with intervals without a "begin" or "end", i.e., possible incomplete tasks. The intervals found are extracted from the log. 3) Extract executed tasks – For each interval found, the algorithm applies a LCS (Longest Common Subsequence) [14] function to measure how similar the extracted interval is related to the expected sequence for the referred task. If the interval has a certain similarity rate, the sequence is extracted from the log.

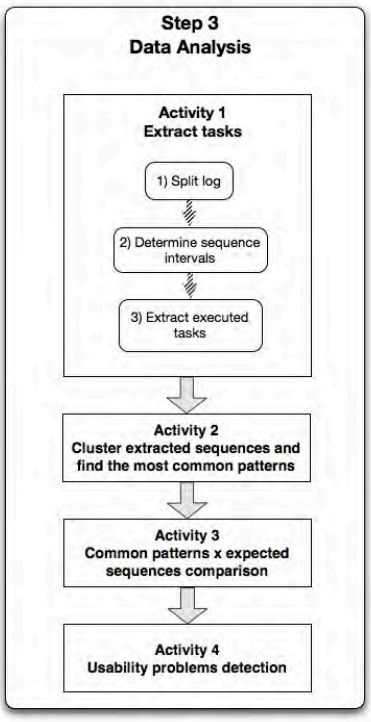

Figure 2. Data analysis.

The second activity in the analysis is the clustering of the extracted sequences and the searching for the most common patterns for the task, i.e. the most common way users performed the referred task.

In the third activity, the expected sequence for the referred task is compared with the most common patterns of execution for the task. Looking for the differences between the "expected" execution and the "actual" patterns is possible to identify if the task is really performed as planned by designers or not. Moreover, it is possible to find problems in the execution of the task and usability problems in the interface of the application.

In the fourth activity heuristics are used to detect problems. The heuristics are sequences of actions that represent known usability problems. If in one pattern found has a match of actions with any heuristic, a possible problem can be detected.

In the end of the analysis, the evaluator obtains the results composed by: the most common patterns of execution for each task; differences between the "expected" execution and the "actual" patterns found; and hints of usability problems detected.

Considerations About the Proposed Method

Our proposed method has some advantages over the tools presented in the related work section of this paper. In Step 1 of WebHint, we tried to simplify the task definition avoiding the use of notations. Opposite to WebRemUsine and WAUTER using notations, our method just requires a simple execution of the task in the application's interface to define it.

In Step 2, we monitor users behavior capturing all actions performed in the interface of the web application evaluated. We intend to find the most common patterns of actions executed by users performing the tasks as in WebQuilt's approach. However, our analysis is deeper than in WebQuilt and AWUSA due to our captured data being more accurate, including mouse movements, clicks and all actions performed in a webpage, not only the sequence of pages requested. Using a proxy approach as UsaProxy we do not have the workload of manually editing each webpage in order to insert code to capture users interaction as in WebVip, WET, WebRemUsine and WELFIT. It is also not necessary to install software on a client's computer as in WAUTER or WebQuilt.

In Step 3 of WebHint, we analyze all actions from all users interacting with the application. It allows us to make a quantitative analysis of the usability of the application. Differently than WebRemUsine and WAUTER, just performing the comparison between two single sequences of actions, our analysis uses the most common patterns found to compare to the expected sequence. It gives us a comprehensive analysis of the users behavior. Finally, in our method we intend to use heuristics in order to automate the usability problems detection.

METHOD APPLICATION

In order to validate our method, a first experiment was carried out. In the experiment, 52 users had their interaction monitored for a period of 2 months in TelEduc[1] (beta version 4.1.1). TelEduc is an environment for on-line courses where users have tools to interact with, as mailbox, file repository, wall, etc. A simulated course was prepared in TelEduc for the experiment and the users were invited to perform some tasks as participants of the course. The experiment had the goal to be a pilot test for the method.

In the experiment we used the UsaProxy tool for capturing the users interaction in Step 2 and for capturing the task definition in Step 1. In Step 3 we used some implemented algorithms for extracting the tasks from the log, as mentioned previously in the Proposed Method section. The process-mining tool ProM [1] [7] was also used for clustering and detecting the execution patterns of the tasks.

RESULTS

In this pilot experiment WebHint showed good potential in finding usability problems, as described in the example below. The task analyzed in the example consists in: "reply an email message received", using the tool Mail of TelEduc. One of the interaction patterns found in the users behavior for this task is illustrated in Figure 3.

[1] http://www.teleduc.org.br

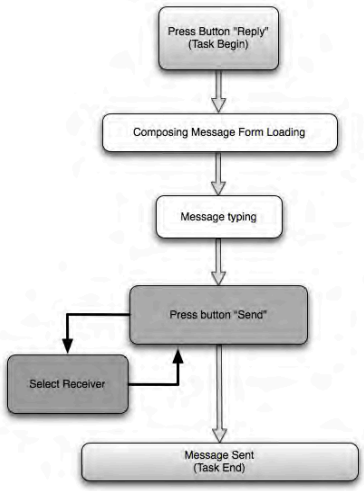

Figure 3. Example of usability problem detected in the experiment.

The loop in the diagram shows that users perform the following steps: “press the button Send“, “select the receiver” and finally “press the button Send” again to finish the task. This behavior differs from the expected sequence of actions for the task. The expected sequence consists of “pressing the button Send” once to finish the task. There is no selecting of the receiver. Analyzing this pattern found it was possible to figure out that the interface of the application does not set the receiver of the message as default, when the button “Reply“ is pressed. So, it is always necessary to set the receiver of the message, even if the user is just performing a reply message action. This is a usability problem because the application breaks an interface standard for e-mail tools, which consists in setting the previous sender as the receiver for the reply message.

In the analysis of the data from the experiment we did not use heuristics to automatic detection of usability problems. The heuristics for problems detection are still in development. The usability problems, as the one illustrated in the example above, were detected analyzing the results obtained from the sequences clustering, the patterns of usage found, and the patterns comparison performed using the ProM tool.

The pilot experiment developed with WebHint, even executed with a small number of users, achieved its goal, being useful to validate the method. The results of the experiment point to the potential of our method in performing remote and semi-automatic usability evaluations based on users behavior analysis.

REFERENCES

1. Van der Aalst, W.M.P., Weijters, A.J.M.M. 2004. Process mining: a research agenda. Computers in Industry 53, 231–244.
2. Andreasen, M. S., Nielsen, H. V., Schrøder, S. O., and Stage, J. 2007. What happened to remote usability testing?: an empirical study of three methods. In Proc. of the SIGCHI - CHI '07. ACM, New York, NY, 1405-1414.
3. Atterer, R. 2006. Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy. In Proc. of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, Edinburgh, Scotland, May 2006.
4. Baker, S.; Au, F.; Dobbie, G.; Warren, I. 2008. Automated Usability Testing Using HUI Analyzer. In ASWEC 2008, vol., no., pp.579-588, 26-28 March 2008
5. Balbo, S, Goschnick, S, Tong, D, Paris, C. 2005. Leading Web Usability Evaluations to WAUTER. In The Eleventh Australasian World Wide Web Conference, Gold Coast, 2005.
6. Cato, J. User-Centred Web Design, Addison Wesley, 2001.
7. van Dongen, B.F., Alves de Medeiros, B.F., Verbeek, B.F., Weijters, A.J.M.M. and van der Aalst, W.M.P. 2005. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, Application and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science, pages 444–454. Springer-Verlag, Berlin, 2005.
8. Etgen, M. and Cantor, J. 1999. What does getting WET (web event-logging tool) mean for web usability. In Proc. of the Fifth Conference on Human Factors & the Web (Gaithersburg, MD, 1999).
9. Hong, J, I., Heer, J., Waterson, S., Landay, J,A. 2001. WebQuilt: A proxy-based approach to remote web usability testing. ACM Trans. Inf. Syst. 19, 3 Jul. 2001.
10. López J. M., Fajardo I., Abascal J.2007 Towards Remote Empirical Evaluation of Web Pages' Usability. In Jacko J. A.(Ed.): Human-Computer Interaction. Interaction Design and Usability. Part I. LNCS 4550
11. Nielsen, J. Usability Engineering, Academic Press, Boston, MA, 1993.
12. Obendorf, H., Weinreich, H., and Hass, T. 2004. Automatic support for web user studies with SCONE and TEA. In CHI '04. ACM, New York, NY, 1135-1138.
13. Paganelli, L , Paternò, F. 2002. Intelligent analysis of user interactions with web applications. In Proc. of the 7th international conference on Intelligent user interfaces, 2002, San Francisco, California, USA
14. Paterson, M., and Dancik, V. 1994. Longest Common Subsequences. In B. Rovan I. Privara and P. Ruzicka, editors, 19th MFCS'94, LNCS 841, pages 127-142, Kosice, Slovakia, August 1994. Springer Verlag.
15. Santana, V. F., Baranauskas, M. C. C. 2008. A Prospect of Websites Evaluation Tools Based on Event Logs. In: HCIS 2008 Milan. Springer, 2008. v. 272. p. 99-104.
16. Scholtz, J. AND Laskowski, S. 1998. Developing usability tools and techniques for designing and testing web sites. In Proc. of the Fourth Conference on Human Factors & the Web, Basking Ridge, NJ, Jun, 1998.
17. Tiedtke, T., Märtin, C., Gerth, N. 2002. AWUSA – A Tool for Automated Website Analysis. In: Proc. of the 9th Int. Workshop DSV-IS 2002, Rostock, Germany, pp. 251–266, 2002.