Webjig: An Automated User Data Collection System for Website Usability Evaluation

Mikio Kiura[1], Masao Ohira[1] and Ken-ichi Matsumoto[1],

[1] Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, Japan
{mikio-k, masao, matumoto}@is.naist.jp

Abstract. In order to improve website usability, it is important for developers to understand how users access websites. In this paper, we present Webjig, which is a support system for website usability evaluation in order to resolve the problems associated with the existing systems. Webjig can collect users' interaction data from static and dynamic websites. Moreover, by using Webjig, developers can precisely identify users' activities on websites. By performing an experiment to evaluate the usefulness of Webjig, we have confirmed that developers could effectively improve website usability.

Keywords: web usability, usability evaluation, analysis of user interactions, dynamic websites.

1 Introduction

It has been found that there are various benefits associated with improving website usability, and so, in recent times, there have been increasing interests on website usability. For instance, in the case of an EC (electronic commerce) site, website usability has an impact on conversion rates. In the case of a website used in workplaces, website usability can also affect the work efficiency. In addition, the cost of user support can be reduced by improving website usability. It also influences a company's image.

Developers must understand how users access a website in order to improve website usability [1]. Usability testing is widely used for understanding users' interactions on a website [2]. In usability testing, users perform some specific tasks within a set time under the supervision of usability engineers (or experts). The engineers observe how the users follow certain steps to accomplish the tasks. Through usability testing, developers can presume users' intellectual process and observe their interaction; these developers can identify problems, clarify design issues, or come up with new ideas.

Although usability testing is a prevalent approach for improving website usability, it cannot be applied to every website. Usability testing requires stakeholders (e.g., users, developers, and experts) to spend large amounts of time and money. Developers cannot conduct usability testing easily [3]. So far, several systems [4, 5, 6, and 7] for website usability evaluation have been proposed so that developers can understand how users access a website over a network with low cost.

However, these systems are designed to collect data only from static websites. Developers cannot figure out users' interactions on a dynamic website (e.g., automatically created webpages by CGI or server-side script, and webpages developed by JavaScript in a Web browser). By using JavaScript, developers can implement an interface that can switch the displayed contents by tabs, drop down menus, or drag-and-drop methods without URL transitions. On a website using such interfaces, the existing systems cannot obtain the previously displayed contents accessed by users because these contents would change.

In this paper, we propose Webjig, which is a support system for website usability evaluation for both dynamic and static websites; this system records users' interactions related to the contents that are displayed in users' Web browser. Developers can exactly understand users' interactions on a website by using Webjig. Thus, they can efficiently improve website usability.

2 Related Work

The traditional approach to resolving problems of website usability it to use the Web server accesses logs [4]. Developers can know various kinds of information including users' IP address, accessed time, request data, and Web server's response from the Web server access log. The advantage of using the Web access sever log is that the access log is automatically saved in a Web server and can be used by developers with low cost. If developers can easily use the Web access log to improve website usability, however, they cannot know users' interactions such as mouse motions, mouse-click positions, and mouse-click timings on a website [5].

Several systems have been proposed to automatically collect the data of users' interaction on a website (e.g., MouseTrack [6], UsaProxy [7]). The systems solved the problem above, by identifying users' mouse motions, mouse-click positions, and mouse-click timings by embedding JavaScript codes into a webpage. The systems helped developers understand users' interactions on a website at a considerably low cost.

Previous studies have suggested that there is a correlation between the point of gaze and the position of mouse cursor. Chen et al. have reported that there is a strong correlation between the point of gaze and the position of mouse cursor; further, the developer can predict a point in the website where the user interested in and they may chart a pattern of the user by users' interaction [8]. In addition, Muller et al. reported that 35% of users traced a sentence with a mouse cursor when they read the sentence in a website. These results show that developers can detect the problems of website usability by studying users' interaction on it.

3 Webjig

In this paper, we introduce Webjig, which is a new system used to solve the problems of the existing systems. Webjig can handle data from static and dynamic websites. By analyzing DOM (Document Object Model) of HTML, Webjig can collect the data of contents clicked by users, including timings, positions, and motions. This mechanism allows usability engineers and developers to solve the problems associated with the existing system, i.e., the existing system could not precisely identify users' interactions on a dynamic website.

We present the system architecture of Webjig in Fig.1. Webjig is a client/server system. The client is implemented by using JavaScript, which executes in a Web browser. The server is implemented by using PHP. The system consists of Webjig::Fetch, Webjig::Analysis, and Webjig::DB.

Webjig::Fetch is a subsystem that automatically collects the data of users' interactions on a website. Webjig::Analysis is a subsystem that shows the information of users' interactions to developers. Webjig::DB is a subsystem that holds the data of users' interactions and provides API to access the data.

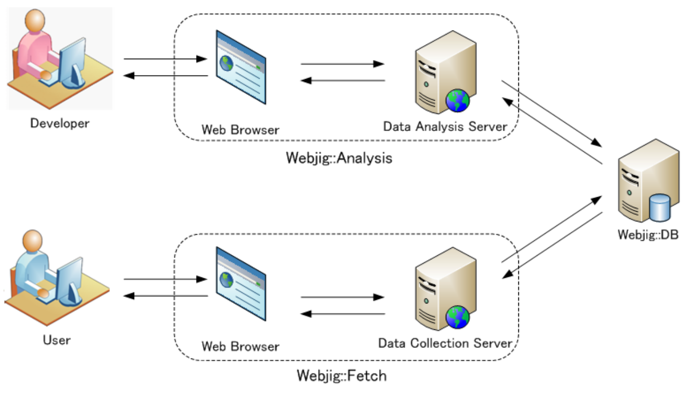

Fig. 1. System architecture of Webjig

3.1 Webjig::Fetch

Webjig::Fetch is a subsystem that automatically collects the data of users' interactions on the website. Table 1 shows the data collected and stored by Webjig. During the time in which a user stays on a webpage, the data may be changed, except for the name and version of the Web browser. The system monitors a change in the data at intervals of dozens of milliseconds and sends the data to Webjig::DB at intervals of few seconds and at the time when the user exits the webpage.

Table 1. Collected data usign Webjig.

Data type	Timing of data collection	Timing of data transmission
Name and version of Web browser	Loaded	Loaded
Inner size of Web browser	Changed	Intervals and exit
Position of scroll bar	Changed	Intervals and exit
Position of mouse cursor	Changed	Intervals and exit
Timing and type of mouse click	Pressed	Intervals and exit
Timing and type of key pressed	Pressed	Intervals and exit
Contents displayed in a Web browser	Changed	Intervals and exit

For collecting users' interactions data, developers have to install Webjig::Fetch in a webpage. what developers have to do is only to insert a line **<script src="URL of Webjig::Fetch" ></script>** in the HTML source code of the webpage that targets the usability evaluation using Webjig. Fig.2 is an example of Webjig installed in an HTML source code. Webjig works even if the developer may insert the script tag at the any place in the HTML source code. However, a mainstream Web browser interprets the HTML source code from the top and displays the contents. Therefore, we recommend inserting the script tag at the bottom of the HTML source code so that Webjig does not disturb the original contents.

```
<html>
<head>
<title>Sample Page<title>
</head>
<body>
<p>Sample Content</p>
<script src="http://example.com/webjig.js" ></script>
</body>
</html>
```

Fig. 2. An example of HTML source code

3.2 Webjig::Analysis

Webjig::Analysis has various features for supporting website usability evaluation. For instance, Webjig::Analysis can replay users' interactions such as mouse motions, mouse click, and keyboard input related to the displayed contents in a movie format by using the collected data.

In Fig.3, we show a screenshot of Webjig::Analysis when it replays the users' interactions. The system consists of displayed contents in a Web browser and some floating windows that control the system and show various kinds of information. Developers can replay users' interactions such as play, stop, forward, and rewind anytime by using various control buttons, seek bar, or slider available on the control window. In addition, the system can also generate a heat map, which shows where the users often click, and presume the portions where the users read and do not read on a webpage. By using these features, developers can examine the following questions.

- Are there any confusing graphics in links?
- Do users pay attention to the content that developers want them to read?
- Where do users look or not look?
- How do users access the website?
- What do user wrong operation on the way to the goal?
- How do users use a dynamic interface?
- Where do users pause when they input into forms?
- Where did the user view before exiting the website?
- and so forth.

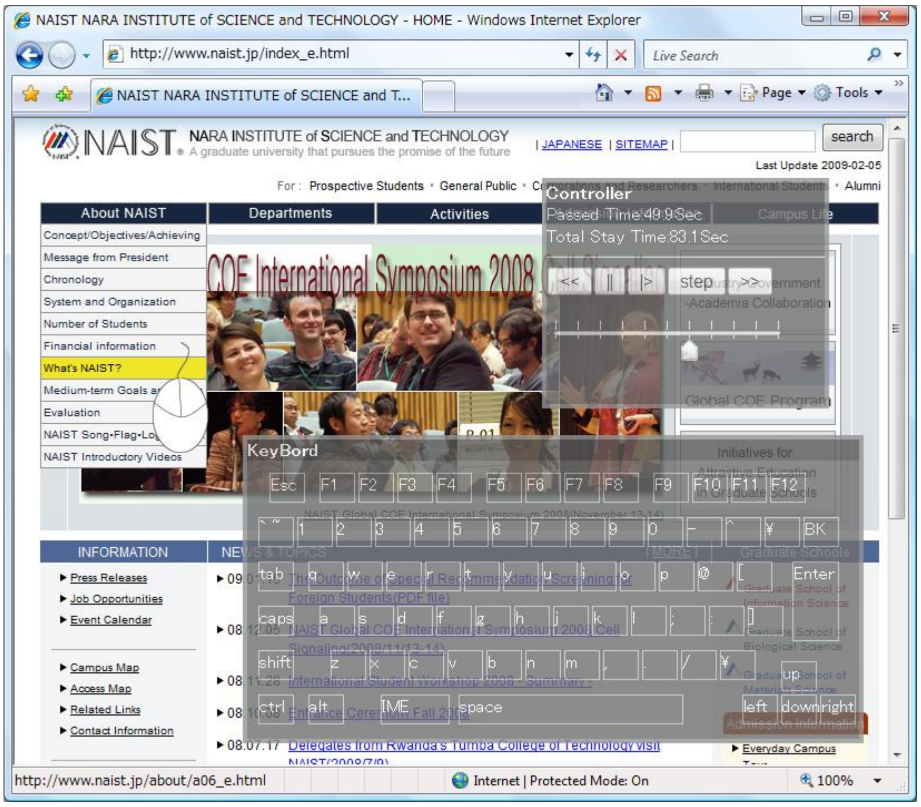
NAIST NARA INSTITUTE of SCIENCE and TECHNOLOGY - HOME - Windows Internet Explorer
http://www.naist.jp/index_e.html
Controller
Passed Time:49.9Sec
Total Stay Time:83.1Sec
<<
||
|>
step
>>
KeyBord
Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
tab
caps
shift
ctrl
alt
IME
space
Enter
BK
up
left
down
right
http://www.naist.jp/about/a06_e.html
Internet | Protected Mode: On
100%

Fig. 3. Screenshot of Webjig::Analysis.

4 System Evaluation

4.1 Overview

We performed an experiment to evaluate the usefulness of Webjig. 54 graduate students (39 males and 15 females, average age 20) participated in the experiment as subjects. 54 subjects were divided into three groups. Each group worked on different tasks described in the next subsection.

4.2 Experiment procedure and task

We executed the experiment according to the following procedures.

Step 1. We provided 24 uses (subject of Group A) five tasks. Each task required the subjects to find a specified product from a dynamic menu implemented using JavaScript. Webjig recorded users' interactions during task execution.

Step 2. Based on the collected data in Step 1, three subjects who had a role of developers (Group B) analyzed the users' interactions during task execution using Webjig::Analysis. The developers planned for an improved structure of menu.

Step 3. We provided 27 different users (subjects of Group C) tasks similar to Step 1. The difference between Step 1 and Step 2 is that the subject of Group C used the improved menu. Webjig recorded users' interactions during task execution.

Step 4. Finally, comparing the task execution time of Step1 and Step3, we checked the validity of the change in the structure of the menu.

Fig.4 is the dummy website for the experiment. Table 1 shows target products and categories where the products exist.

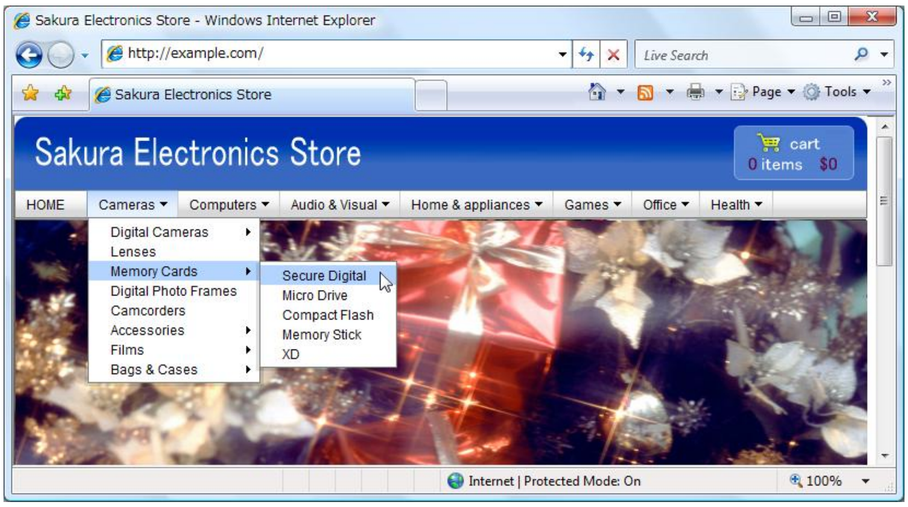

Fig. 4. Screenshot of the dummy website for the experiment.

Table 2. Target products and category for each task.

Task Name	Product	Category
Task 1	Dry cell	Audio & visual
Task 2	SD memory card	Cameras
Task 3	A massage chair	Health
Task 4	Electronic dictionary	Office
Task 5	Fax	House & appliance

4.3 Experiment results

Developers can know where users look in the webpage by using Webjig. Table 3 shows what percentage of the subject of Group A firstly clicked on which categories. The grayed rectangle in Table 3 means the correct category where a specified product exists for each task. For example, 54% of the subjects first clicked on the category of house & appliance, thought dry cell belonged to the category of audio & visual. When using existing systems, developers cannot know such the information.

Table 3. Results of first category sellection.

Category	Task 1	Task 2	Task 3	Task 4	task5
Camera	29%	13%	0%	0%	0%
Computer	0%	46%	0%	13%	4%
Audio-video equipment	4%	33%	0%	0%	21%
House & appliance	54%	4%	71%	29%	29%
Game	4%	4%	0%	0%	0%
Office equipment	8%	0%	4%	58%	46%
Health	0%	0%	25%	0%	0%

Table 4 shows the changed structure of the menu which was planned by the developers based on the result of Table 3. The plan is made from an idea that if there was the category more clicked by users than the current category, a target product should be moved to a proper category.

In case of task 1 where subjects searched a dry cell, a dry cell belonged to the audio & visual category, but many subjects first pay attention to the house & appliance category. Therefore, the developers moved the dry cell to the category of house & appliance. Further, in case of task 4 where subjects searched an electronic dictionary, an electronic dictionary belonged to the category of office equipment, and the majority of the subjects first paid attention to the office equipment category. Therefore, the developers did not move it to any other category.

Table 4. Change plan for the menu of the categories.

Task Name	Product	Original category	Destination category
Task 1	Dry cell	Audio & video	House & appliance
Task 2	SD Memory Card	Cameras	Computers
Task 3	A massage chair	Health	House & appliance
Task 4	Electronic dictionary	Office	Office
Task 5	Fax	House & appliance	Office

We perform the experiment after changing the website, as shown in table 7. We show the experiment result in Fig.5. From Fig.5, the task execution time has been reduced in tasks 1, 2, and 3 by applying the changed plan.

Fig. 5 shows the results of the execution time for each task in Step1 and Step 3[1].We can confirm that the execution time in Step 3 is shorter than that in Step 1, that is, the improved menu structure based on the developers' analysis using Wegjig was effective.

[1] Since the structure of the menu was changed in Task 4,we could not confirm the significant difference between the results in Step1 and Step3.

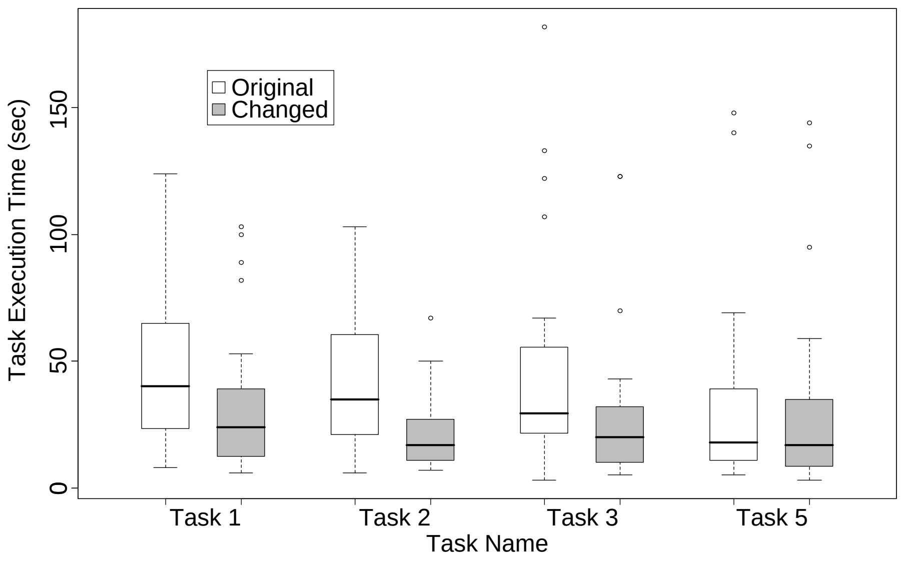

Fig. 5. Result of the task execution time in Step1 and Step3.

5 Discussion

By using Webjig, developers can obtain information which they would not have got with the existing systems. For this reason, developers can detect problems in website usability and create a plan for improving website usability by collecting data of users' interactions, as performed in this experiment.

In the experiment where users choose the items from the menu, the developers can determine the execution time for each task by using existing systems. Thus, they can detect the problems of usability by comparing the execution time of each task and pinpoint the task where the execution time is longer than that taken by another task. In Fig.5, the execution time of tasks 1, 2, 3, and 5 is longer than that of task 4. For this reason, a developer can hypothesize that there remains problems of website usability. However, it is difficult to eliminate the problem if they cannot understand the cause of the problem.

By using Webjig, a developer can efficiently detect the problem of website usability. In case of task 1 (subjects find a dry cell), we show the experiment result in table 3; dry cell belongs to audio-visual equipment, but many subjects pay attention to household appliance. The developer hypothesized that "Many users think that a dry cell belongs to a household appliance" and moved the dry cell from audio-visual equipment to household appliance. As a result, the execution time is reduced before changing the category.

According to Fig.5, the task execution time of the changed website is less than that of the original website. In tasks 1, 2, and 3, we can observe significant improvement in the execution time. However, in task 5, we did not observe any significant improvement in the execution time.

Table 5. Priority for the improvement

Task Name	Correct category (A)	Current Category (B)	B/A
Task 1	4%	54%	13.5
Task 2	13%	46%	3.5
Task 3	25%	71%	2.8
Task 5	29%	46%	1.6

We explain the reason for this. In table 5, we compare the rate of users who pay attention to the correct category with the rate of users who pay attention to the changed category. In case of task 1, 4% of users pay attention to the correct category (a category of audio & visual) when searching for dry cell and 54% of users pay attention to the wrong category (a category of house & appliance) when searching for dry cell. This has a difference of 13.5 times. Similarly, task 2 has a difference of 3.5 times, task 3 has a difference of 2.8 times, and task 5 has a difference of 1.6 times. As a result, we can say that if there is not a big difference in the rate of users who pay attention to an original category and the rate of users who pay attention to a changed category, we cannot confirm an effect in the change.

Therefore, developers have to examine whether the usability is improved by understanding users' interactions and not by the reason that the task execution time was longer than others. By using Webjig, a developer can exactly understand users' interactions and examine whether the usability is improved. However, it is difficult to examine the improvement of website usability by using existing systems because exact users' interactions cannot be obtained.

However, developers cannot use the Webjig instead of user testing because they can know the gaze point by using the eye tracking system and they can know the intention of the user by interviewing him/her during user testing. But we saw that there was the point that could be improved website usability by using Webjig. Therefore, developers may efficiently improve website usability by combining user testing and Webjig.

6 Conclusion and future work

In this paper, we proposed a Webjig support system for static and dynamic websites. As a result of the experiment, we show that developers can improve website usability effectively by using Webjig. In the future, we are going to think about the cost of website usability evaluation between existing systems and Webjig and compare usability testing with Webjig to determine the efficiency of website usability evaluation.

Acknowledgements

This study is supported by Information-technology Promotion Agency, Japan (IPA), Exploratory IT Human Resources Project (MITOU Program) in the fiscal year 2008.

References

1. Nielsen, J. and Landauer, T.K.: A mathematical model of the finding of usability problems, *the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pp.206–213 (1993)
2. Dumas, J.S. and Redish, J.C.: *A Practical Guide to Usability Testing*, Norwood, New Jersey, Ablex Publishing (1993)
3. Barnum, C. M.: *Usability Testing and Research*, Longman, London (2001)
4. Hong, J.I. and Landay, J.A.: WebQuilt: a framework for capturing and visualizing the web experience, the 10th international conference on World Wide Web (WWW '01), pp.717–724 (2001)
5. Etgan, M. and Cantoe, J.: What does getting WET (Web Event-logging Tool) mean for web usability?, 5th Conference on Human Factors and the Web(HFWEB99), available from http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html accessed 2009-2-27 (1999)
6. Arroyo, E., Selker, T. and Wei, W.: Usability tool for analysis of web designs using mouse tracks, CHI'06 extended abstracts on Human factors in computing systems, pp.484–489 (2006)
7. Atterer, R. and Schmidt, A.: Tracking the interaction of users with AJAX applications for usability testing, the SIGCHI conference on Human factors in computing systems (CHI '07), pp.1347–1350(2007)
8. Chen, M.C., Anderson, J.R. and Sohn, M.H.: What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing, *CHI '01 extended abstracts on Human factors in computing systems*, pp.281–282 (2001)

9 Mueller, F. and Lockerd, A.: Cheese: tracking mouse movement activity on websites, a tool for user modeling, *CHI '01 extended abstracts on Human factors in computing systems*, pp.279–280 (2001)