J. King Saud Univ., Vol. 20, *Comp. & Info. Sci.*, pp. 1-11, Riyadh (2008/1429H.)

RWELS: A Remote Web Event Logging System

I. Shah, L. Al-Toaimy* and M. Jawed
Department of Computer Science,
College of Computer & Information Sciences,
King Saud University, Riyadh, Saudi Arabia

(Received 09/09/2006; accepted for publication 02/07/2007)

Abstract. Event logs are an important data source for identifying usability problems in websites. We present a web-based client-server application, Remote Web Event Logging System (RWELS), for logging user-interface events generated in Microsoft Internet Explorer during a user's interaction with the pages of a website. RWELS logs events without interfering with the user's interaction—no additional interaction is required on the part of a user to enable logging. RWELS is configurable and allows user-centric event logging. A usability analyst can choose the set of events to be captured and the pages of the website to be logged for a particular user. The event logs are dispatched through HTTP to the server where they are stored as text files. Users are identified uniquely and the event logs are associated with the user sessions.

1. Introduction

Web event logs are an important data source for usability evaluation. Web event logs can be analyzed using various techniques to identify usability problems in websites. Event-logging tools capture events on the server-side, in the proxy or client-side. This paper describes Remote Web Event Logging System (RWELS), a client-server application that logs events generated in Microsoft Internet Explorer as a result of users' interaction with the pages of a website and dispatches the log to the server.

RWELS can capture user-interaction events such as mouse clicks, page scrolls, mouse moves and key-presses. It abstracts out important event information and filters out unnecessary low-level detail from the raw event data. For example, for the mouse move events, it records the start and end positions of the mouse movement only, and for scrolling events it computes the distance scrolled in pixels. This preprocessing reduces the amount of data captured and reduces the bandwidth requirements for transferring data to the server.

There are a few issues with regard to existing web event logging systems that RWELS attempts to address. One of these issues concerns obtrusiveness of the logging technique. When event data is to be captured for usability evaluation of a website, the logging process should not interfere with a user's interaction with the web page, otherwise evaluation results obtained can be biased. Generally, existing web event logging systems do not satisfy this requirement. They are obtrusive and require users to perform additional interaction to enable logging. RWELS uses JavaScript event-handler functions to silently capture and log events while a user browses a website. The event log is automatically dispatched to the server, where the logs are stored as text files. A user does not have to intervene anywhere in this process.

Configurability is another important issue in event logging systems. When event data is to be captured on a large scale from the users of a website, the logging tool must be configurable. It should be possible to capture the event data on "as needed" basis from only the users a usability analyst is interested in. RWELS satisfies this requirement. It maintains on the server a database of users and for each user a profile is maintained that indicates the set of events to be logged for the user. Whenever a page request arrives, the users' profile is retrieved from the database and in accordance with the profile JavaScript event capturing code is enabled dynamically and inserted in the requested page (Fig. 1).

[*] The author is with SAMBA, Riyadh.

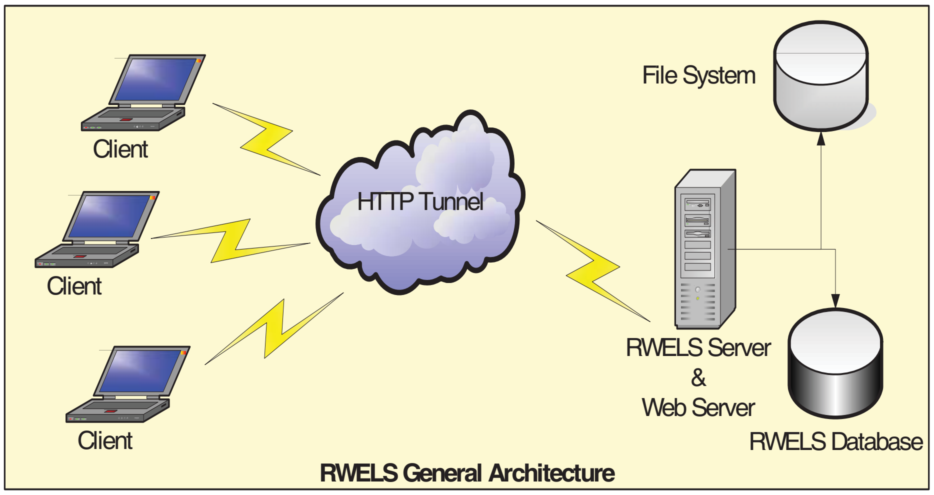


Fig. 1. The overall architecture of RWELS.

Section 2 presents the background information and relates web event logging to usability evaluation. A classification of web event logging techniques is given. Various issues related to web event logging techniques and how they are dealt with in RWELS is also discussed. Section 3 describes RWELS architecture, event capturing mechanism and RWELS testing and future development. Section 4 describes some client-sided event logging tools related to RWELS. Section 5 concludes the paper.

2. Web Event Logging Techniques

Usability evaluation is about measuring the user friendliness of a system's user interface and identifying usability problems in it (Dix *et al.*, 2004). During usability evaluation, the following activities are carried out: Capture – data concerning the user interface is captured, Analysis – the captured data is analyzed, and Critique – usability problems are identified in the interface. Event logging is a data capturing technique used in some usability evaluation techniques. Events are generated as a result of users performing physical actions with various input devices during interaction with event-driven applications. Event logs record the user-interface events, their time and order of occurrence and the interface objects manipulated in each action and thus provide a more or less complete record of the actions that a user performs at the interface. Event logs can be recorded automatically and processed to obtain usability information such as the task completion times, task errors, etc. (Hilbert *et al.*, 2000; Nielson, 1993).

An example of an event log captured with RWELS appears in Fig. 2. This is a log of a user browsing a news page. The scrolling events indicate the user scrolling down the page *steadily* and probably reading the text. In contrast to this, an event log containing many quick up/down scrolls would have indicated the user searching for information, possibly due to a usability problem. Analysis of the scroll events can be visualized as scroll-time plots shown in Fig. 3.

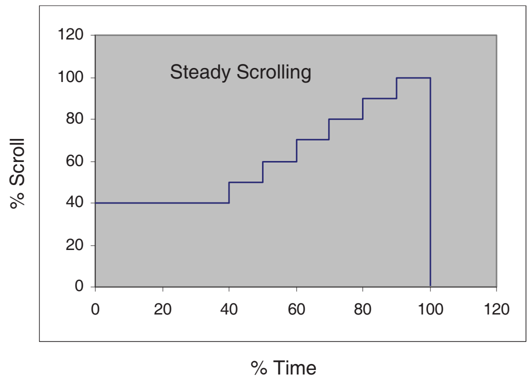
The rest of this section presents a classification of the web event logging techniques and various issues related to web event logging.

2.1. Classification

The event logging techniques have been classified into various categories based on whether a technique is manual or automatic and the point where event data is accessed for logging (Shah and Jawed, 2006). *Manual logging* involves observing a user's interaction with the application's user interface directly in a laboratory or through a video recording, and noting down the actions performed by the user (Byrne *et al.*, 1999). *Automatic logging* records user interaction events through instrumentation code embedded within the application code or through a software component independent of the application and running in the background, e.g. (Choo *et al.*, n.d.; Ellis *et al.*, 1998; Etgen and Cantor, 1999; Hammontree *et al.*, 1992; Hartson *et al.*, 1996). Automatic logging can take place in a laboratory setting where test subjects operate instrumented computers and/or application code in a controlled environment, or remotely, with users working in their natural work setting. Remote logging is suited to web applications but can be used with other applications as well. Remote web event logging can take place on the client-side as the *client-side logging*, e.g. (Etgen and Cantor, 1999; Fenstermacher and Ginsburg, 2003; Gamboa and Ferreira, 2003; Goecks and Shavlik, 2000; Gonzalez Rodriguez, 2000; Hilbert *et al.*, 1998; Laus, 2001; Paganelli and Paterno, 2002; Reeder *et al.*, 2000, Scholtz *et al.*, 1998, Shahabi and Banaei-Kashani, 2002), in the proxy as *proxy-based logging* (Hong and Landay, 2001) or on the server side as *server-side logging*, e.g. (Drott, 1998; Gil, 2004; Helms *et al.*, 2000; Marques *et al.*, 2004; Perkowitz and Etzioni, 2000; Spiliopoulou and Pohle, 2000; Spiliopoulou, 2000; Tiedtke *et al.*, 2002). On the client-side, logging can take place through *custom developed software* installed and running as a separate component in the background, e.g. (Choo *et al.*, n.d.; Choo *et al.*, 1999; Hilbert *et al.*, 1998), through custom build *instrumented browsers*, e.g. (Fenstermacher and Ginsburg, 2003; Goecks and Shalvik, 2000; Reeder *et al.*, 2000) or through *scripting code/applets embedded* within the web pages, e.g. (Etgen and Cantor, 1999; Gamboa and Ferreira, 2003; Laus, 2001; Paganelli and Paterno, 2002; Scholtz *et al.*, 1998; Shahabi and Banaei-Kashani, 2002). RWELS belongs to this category of client-side remote event logging techniques.

EVENT NAME		TIME (msec)	READ TIME (s)
PAGE LOADED	news.bbc.co.uk/m_e/658.stm	1727100	
MOUSE MOVE	566,318)(892,180)	1728881	
SCROLL DOWN	0 182 1004 613 MWheel	1734288	READ END: 7
MOUSE MOVE	(892,177)(896,177)	1735836	
SCROLL DOWN	0 364 1004 613 MWheel	1772217	READ END: 38
SCROLL DOWN	0 546 1004 613 MWheel	1825601	READ END: 53
SCROLL DOWN	0 637 1004 613 MWheel	1864061	READ END: 38
SCROLL DOWN	0 819 1004 613 MWheel	1871109	READ END: 7
SCROLL DOWN	0 910 1004 613 MWheel	1900005	READ END: 29
SCROLL DOWN	0 1092 1004 613 MWheel	1914304	READ END: 14
MOUSE MOVE	(898,176)(898,176)	1921430	
SCROLL DOWN	0 1365 1004 613 MWheel	1939433	READ END: 25
SCROLL DOWN	0 1456 1004 613 MWheel	1989439	READ END: 50
SCROLL DOWN	0 1638 1004 61 MWheel	2012706	READ END: 23
SCROLL DOWN	0 1820 1004 613 MWheel	2054896	READ END: 42
SCROLL DOWN	0 1911 1004 613 MWheel	2071382	READ END: 16
MOUSE MOVE	(899,176)(899,176)	2083804	
SCROLL DOWN	0 2093 1004 613 MWheel	2084351	READ END: 13
SCROLL DOWN	0 2162 1004 613 MWheel	2109978	READ END: 25
MOUSE MOVE	(899,170)(181,0)	2130026	
PAGE EXITED	404	2131338	READ END: 21

Fig. 2. An event log showing "Steady Scroll Down" pattern indicating that the user is reading the contents of the page. The second number after "SCROLL DOWN" is the amount scrolled in pixels, while as the number after "READ END" is the time spend on the page before the next scroll.



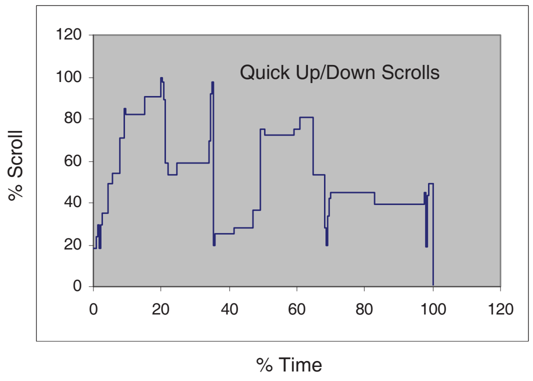


Fig. 3. Typical scroll-time plots for "Steady Scroll Down" and "Quick Up/Down Scroll" patterns.

2.2. Issues and goals

With regard to web event logging systems in general and client-side web event logging techniques in particular, there are some issues that arise and existing logging systems address in different ways. These issues are:

Events captured: A web-event logging technique should be able to capture any event generated to get complete information about the actions users perform at the interface to achieve task goals. This is essential for a comprehensive and successful usability evaluation. Most web-event logging techniques are not able to capture all the generated events mainly due to the security restrictions of client systems.

Browser events can be classified as: the *document events*, resulting from users' mouse-clicks, mouse movements, scrolling events and key-press events in the browser document window; the *application events*, resulting from, for example, when the browser starts or completes downloading a document; and the *interface events*, resulting from users' browser menu selections, clicks on the browser toolbar, and, for example, the backward/forward button clicks. Document and interface events together are usually called the *user interaction* (*UI*) events. Client-side logging techniques that use custom developed software (Choo *et al.*, n.d.; Hilbert *et al.*, 1998), or custom built instrumented browsers (Fenstermacher and Ginsburg, 2003; Goecks and Shalvik, 2000; Reeder *et al.*, 2000) can capture almost all the above types of events, but the techniques employing scripting code can capture only the document events (Etgen and Cantor, 1999; Gamboa and Ferreira, 2003; Laus, 2001; Paganelli and Paterno, 2002; Scholtz *et al.*, 1998; Shahabi and Banaei-Kashani, 2002). The browser's security settings do not allow scripting code to capture browser interface events or the application events.

With regard to the ability to capture events, the goal for RWELS has been the ability to capture as many events as possible. However, the use of scripting code to capture events inherently restricted RWELS's ability to capture events to only document events.

Unobtrusive capture: A web event logging technique should be unobtrusive. The user's interaction with an application should proceed naturally during event logging and ideally the user should be unaware of it. Some web-event logging techniques are highly obtrusive. They require the users to install logging software, respond to questions originating from the logging code or report "critical incidents", e.g. (Choo *et al.*, n.d.; Goecks and Shalvik, 2000; Hilbert *et al.*, 1998). The additional interaction interferes with the users' interaction with the application and may bias the evaluation results. Client-side web-event logging techniques that use custom software and custom browsers are obtrusive, as they require users to install new software on their systems to enable logging. Client-side web-event logging techniques that use scripting code are mostly unobtrusive since the logging code is embedded within the web pages.

RWELS captures events unobtrusively by using JavaScript event handlers embedded in the pages of the website. Users simply access the website and start browsing pages of the website. Event-handlers automatically log the events generated and dispatch the logs to the server without user intervention at any stage.

Configurability: A web-event logging system should be configurable, enabling the usability analyst to capture only the events that he/she is interested in, on "as needed" basis and for only the users he/she is interested in, e.g. (Hilbert *et al.*, 1998). Capturing all the events generated for all the users can result in a deluge of data and in most situations it is not required. For example, if the usability analyst is interested in studying scrolling behavior of users on certain web pages, as in the previous example, capturing events other than the scroll events is unnecessary.

RWELS can be configured to capture any of the available document events. There is a default set of events that RWELS captures for the users who are chosen for logging, but the usability analyst can modify the default set for a user by modifying his/her profile in a server database.

Storage and retrieval: A client-side web event logging system should be able to retrieve event logs with minimum bandwidth requirements. The amount of event data generated through client-side logging can be huge. Systems that monitor and display user interaction in real-time send each individual event to the machine where the interaction is being monitored (Gamboa and Ferreira, 2003). Over the internet this can generate huge network traffic and consume significant amount of bandwidth. Client-side event logging systems usually store captured data temporarily in the client and then send it to a collection point at suitable intervals (Hilbert *et al.*, 1998; Kangas and Chiu, 2001). Two mechanisms are in common use: one based on cookies (Etgen and Cantor, 1999) and the other on applets (Paganelli and

Paterno, 2002). Mechanism based on cookies uses HTTP requests to transfer data, but cookies have a storage limitation – a cookie cannot store more than 2 KB of data. Applets based schemes use applet-servlet communication to transfer data, but the ports used in this communication can be blocked. Email is also used to transfer the logged data (Hilbert *et al.*, 1998).

RWELS uses a JavaScript dynamic array to temporarily store event data. The dynamic array is initialized afresh for each page downloaded into the browser. When a user moves to another page or exits the browser, the contents of the array are passed to an applet that uses an HTTP post request to transfer the log to the server. Ports used by HTTP are usually not firewall blocked.

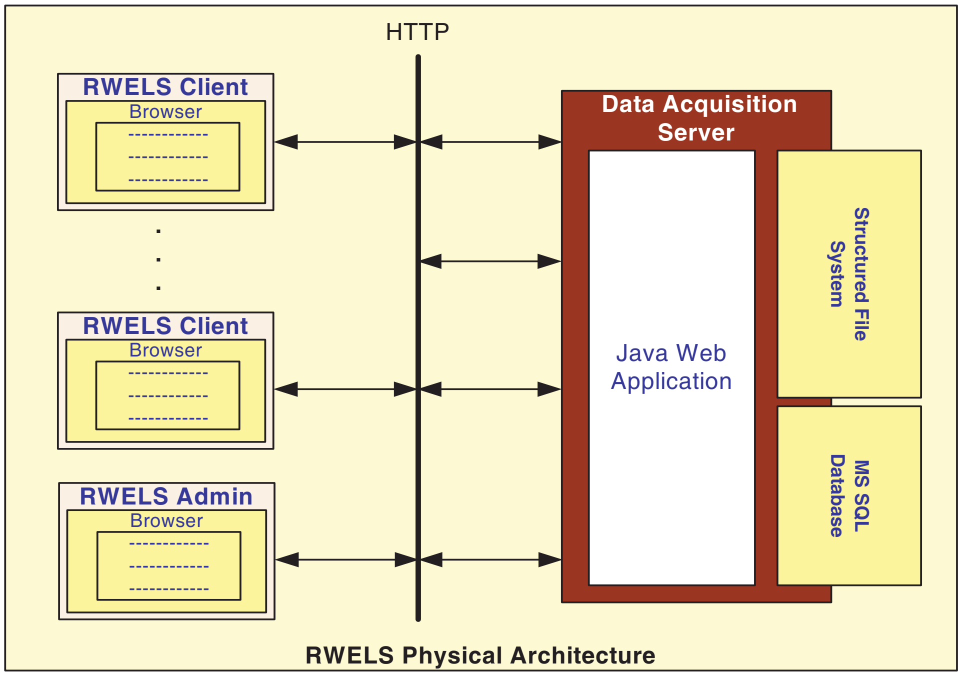


Fig. 4. The physical architecture of RWELS.

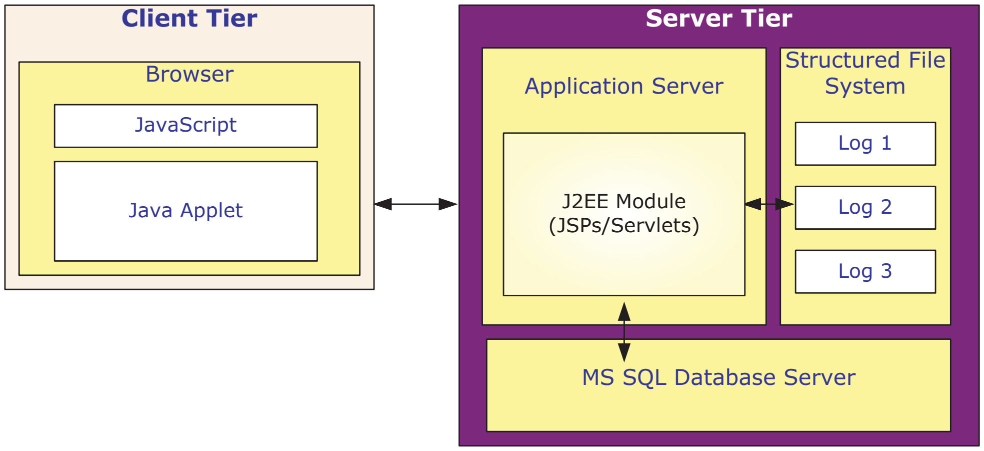


Fig. 5. The client and server side components in RWELS.

3. RWELS System Architecture

RWELS consists of server-side and client-side applications. Server-side application is responsible for acquiring event data from the clients and storing it. Client-side application is responsible for capturing browser events and dispatching them to the server. Usability analyst that is interested in collecting event data for a website interacts with RWELS server-side application. Usability analyst sets up RWELS for the website, creates profiles for users and manages administrator accounts. On the client-side, it is the user who indirectly interacts with RWELS client-side application. A user simply accesses the website and starts interacting with the web pages. The events resulting from the interaction get captured automatically and the event log is dispatched to the server where it is stored. The complete RWELS process is described in the activity diagram in Fig. 6.

3.1. Server-side components

RWELS server-side application is essentially an event data acquisition server consisting of a Java module and an MS SQL database server. The Java module consists of multiple Java Server Pages (JSPs) and Java Servlets. The MS SQL database stores the users' profiles, storing information about the pages and events on each page to be logged for a user. The Java module is responsible for the following:

- When a page request from a user's browser is received, the user's profile from the database is retrieved and JavaScript event handlers in the requested webpage are enabled in accordance with the retrieved profile. A new user is assigned a new ID and registered in the database. The new user's profile is set to its default value.
- A servlet class in the module deals with receiving the event logs from clients and storing them as text files in a file system.
- Servlet classes in the module manage administrator accounts and allow authenticated login to the system. Existing administrators can register a new administrator or un-register an existing administrator. This interaction takes place through a web interface.
- Through other servlet classes in the module, an administrator is able to browse a registered user's profile and modify it. These servlet classes also enable an administrator to edit, delete, rename or download saved logs. This interaction also takes place through a web interface.

The database consists mainly of three tables: WLAdmin, WLClient and WLPage. The WLAdmin holds existing system administrator's usernames and passwords. The WLClient stores existing registered users' event profiles, i.e. their unique IDs and the events to be logged for each user. The WLPage stores existing registered users' page profiles, i.e. their unique IDs along with the pages of the website for which their events are to be logged.

3.2. Client-side components

RWELS client-side application consists of a Java Applet and a JavaScript library of event-handling functions. These get downloaded with the web pages and run in the client browser, capturing and recording events occurring in the browser. The events are stored temporarily in a dynamic array. When a user exits a page (i.e. unload event occurs), the recorded events are passed on to the Java Applet that transfers the event log to the server-side for storage.

Java Applet is a non-GUI applet embedded in the web pages to be event logged. The applet's only function is to transfer event logs from the clients to the server. The transfer is initiated by JavaScript call to the applet through JavaScript *liveconnect* facility. The applet encapsulates event data into Java Serialized Objects and transmits it using HTTP post. Using HTTP has the advantage that HTTP ports are usually not blocked by firewalls.

The events JavaScript library functions capture, depend on a user’s event profile. This information is read from the server-side database table WLClient and the event-handlers corresponding to the events specified in the event profile are enabled. Presently there are event-handlers for capturing and recording mouse clicks, mouse movements (recording starting and ending locations in pixel co-ordinates), scrolling, key-press events and chord key-press events. An example event log is shown in Fig. 2.

For each event that occurs in the browser, event's name, time of occurrence and some additional information is recorded. For the page load events, the URL of the page is recorded. For mouse move events, the start and end point co-ordinates of the movement in pixels are recorded. When scrolling events occur, the scrolling event handler determines whether it was a scroll up or down event and records the event along with the scrolling distance in pixels. Whether a user has scrolled up or down is determined by comparing the document object's current `scrollTop` property value with its value before scrolling. The scrolling distance is determined by computing the difference between the current and previous `scrollTop` values. Whether a user used a mouse wheel in scrolling is determined through the event handler for the mouse wheel event.

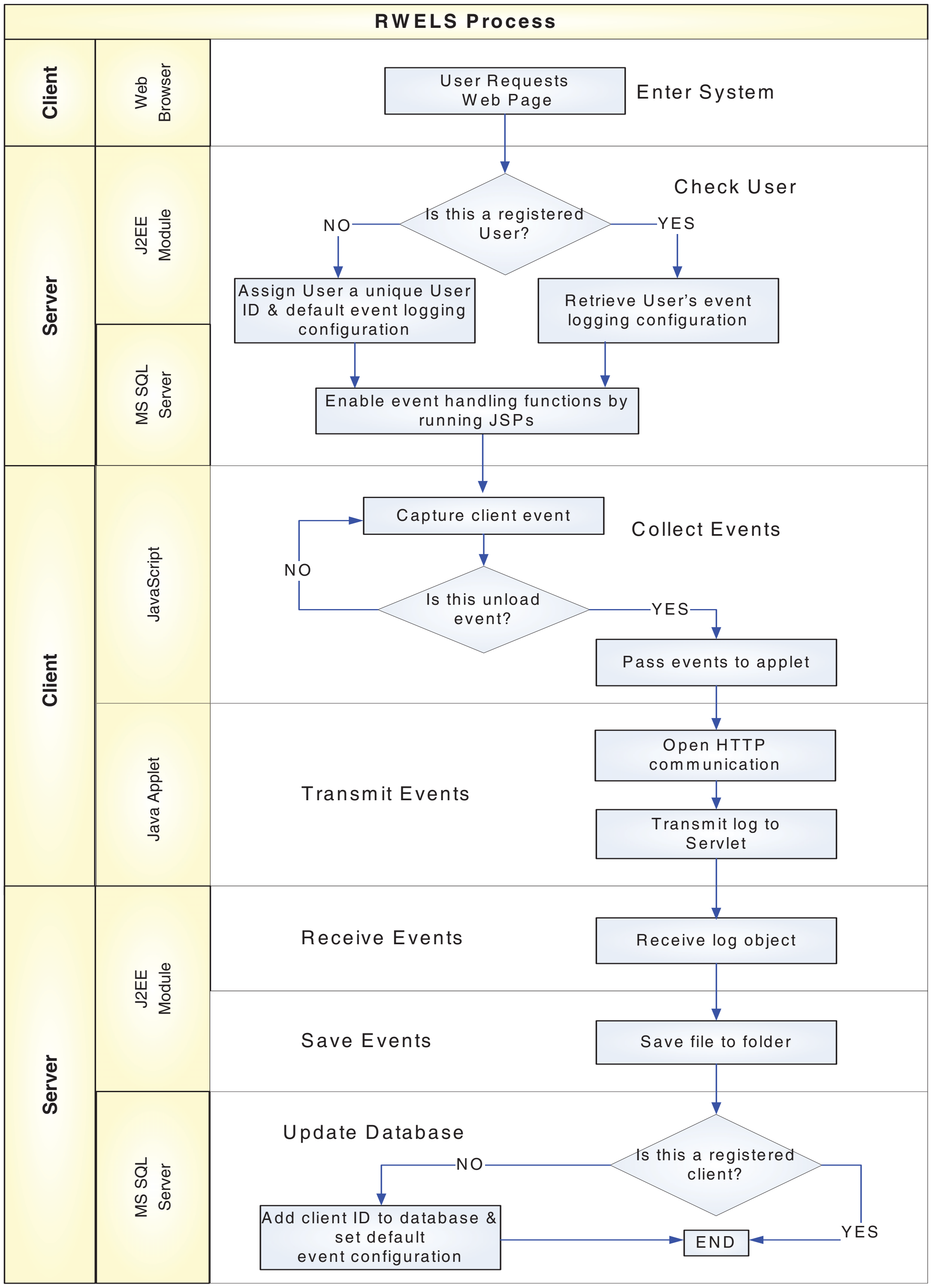


Fig. 6. The RWELS process.

An important piece of data that can be used in drawing conclusions about a user's browsing behavior and a user's interest in a page or section of a page is the time the user spends on the page or section of the page. It is assumed here that this time interval, i.e. the "read time", starts immediately after a page is downloaded and ends when a user scrolls up or down. A timer is used to keep track of the read time. The timer is set using the browser window's `window.setInterval()` method. The timer starts when a page downloads completely and stops when the user starts scrolling. When the scrolling ends, the timer is re-started to register the read time of a new reading interval. Each time scrolling starts, a READ END is logged and the elapsed time is printed out in seconds. Each time scrolling ends a READ START is logged.

Internet Explorer fires a train of closely spaced (~10 ms) events whenever a user scrolls or moves the mouse. Recording all the events in the train can lead to large amount of unnecessary low-level data being captured. Most of the time a usability analyst is interested in the instant of time scrolling begins, i.e. the first scroll event. The recording method used here captures the first scrolling event and ignores all other scroll events that immediately follow it. Similarly, for the mouse move events only the starting mouse move event and ending mouse move event record the start and end points of the mouse movement. In both cases, timers are used that prevent any event capturing in the time interval that starts after the firing of the first event and ends 500 ms after the firing of the last event.

3.3. Testing and future development

RWELS was tested in a local area network environment. It successfully captured event data from users' browsers (Internet Explorer) while the users browsed a website hosted on a local machine running Apache. The captured event logs of several megabytes in size were successfully dispatched to the server and saved as text files. The time delay was of the order of 500 ms for event logs of this size. RWELS is yet to be tested on an internet scale data collection task. Work is in progress to test RWELS for a large website over the internet.

At present, RWELS is set up manually for a website. This involves modifying the pages of a website manually to insert references to files containing various JavaScript functions and applets. It must be ensured that the event-handlers do not interfere with normal page functionality. For a website with a large number of pages, this is a difficult process. An application that sets up RWELS automatically is therefore needed. Such an application would automatically insert references to JavaScript code and applets into the pages of the website.

4. Related Work

There are client-side event logging systems that use embedded JavaScript code and applets to log events and are closely related to RWELS. This section briefly describes four such existing web-event logging systems.

4.1. WET

Web Event-logging Tool (WET) (Etgen and Cantor, 1999) is an event logging system with the aim of studying the usability of the website. WET can record mouse clicks, change, mouseover and page load events in Internet Explorer and Netscape browsers. The usability analysts configure WET manually by specifying the events to be captured and the event handling functions in a JavaScript file called WET.js and inserting a call to the file in the pages of the website. User-centric event logging (i.e. logging is turned on/off for certain users in certain situations) is implemented by encapsulating the call to WET.js within a ColdFusionML tag that checks the users "login name" before writing the call to the WET.js file into the page. Unlike RWELS, WET logs a fixed set of events for all users—and so is not configurable. The logged events are temporarily stored in cookies (with the storage limitation of 2 KB) and retrieved by sending the cookies to the server when they are full. A start/stop button is provided on all pages to allow a user to turn logging on or off. This feature makes logging intrusive.

4.2. WebRemUSINE

WebRemUSINE (Paganelli and Paterno, 2002) is a tool for the usability evaluation of web interfaces. Event data is collected through JavaScript code embedded into the web pages. The script code redefines event handlers to capture the user and browser events, such as: abort and error on images; change on form elements; click and double-click on links, images and form elements; load and unload of pages, submit and reset of forms; resize and scroll of browser windows. From the information provided in Paganelli and Paterno (2002), it appears that the tool is not configurable and logs the same set of events for all users. The logging is obtrusive because the user must indicate the task he/she intends to perform during a site visit and must explicitly indicate the end of a session. When a user enters a website, an applet is launched that remains active for the rest of the session. Event handlers communicate the event data to the applet that temporarily stores the event data and finally dispatches it to the server, where a Servlet inserts time, date and user identification into the log and saves it.

4.3. WIDAM

WIDAM (Gamboa and Ferreira, 2003) is a client-server application that offers many services, including service for monitoring user interaction with web pages in real-time and/or recording it. WIDAM captures mouse movements, mouse clicks, key presses, document unload and various window events through JavaScript. Unlike RWELS, the set of events logged is fixed and not configurable for a user. When a user downloads a page, an applet is launched that creates a socket connection for communicating with the server in real-time. Whenever an event occurs, the event is sent to the server through the socket. There is no temporary storage of the event data in the client. Whenever the server receives an event from the client it stores it in a database and/or passes it to other users observing the interaction with a web page in real-time. Because of this, bandwidth requirements are quite high as compared to RWELS. Furthermore, RWELS uses HTTP instead of a dedicated socket connection. In WIDAM, since the user has to go through a specific web page to select a service, this makes the user aware of the logging.

4.4. WUM

Web Usage Mining (WUM) (Shahabi and Banaei-Kashani, 2002) uses a Java applet to capture user navigation. An applet, embedded in the pages of a website, starts when a page is downloaded into a client browser. The code referencing the applet is inserted automatically in all pages of a website through a simple application. When a user enters a website, the applet is allocated a unique ID generated on the server-side. The information applet is able to capture is meager compared to RWELS and includes just the pages accessed, the time and date a page was loaded into a browser and unloaded, and users' IP address. No other user interaction events are logged. The applet communicates with a data acquisition server. Each time a page is unloaded the logged data is sent to the data acquisition server using TCP. The data acquisition server stores the data in a database. Logging is not user-centric. Logs are recorded as sessions and each session is identified uniquely.

5. Conclusion

RWELS is a client-side web event-logging tool that was designed with the goals of unobtrusive capture of events and configurability in mind. Both these goals have been achieved in the prototype implementation. RWELS is able to capture user-interaction events in Internet Explorer that include mouse clicks, scrolls, mouse moves and key-presses. It dispatches the event-logs to the server where they are stored as text files. A user does not have to intervene in this process and in this way RWELS is unobtrusive. It avoids capturing unnecessary low-level event data and abstracts out important information. This reduces the amount of data to be transferred to the server over the network. To ensure configurability, RWELS maintains a user profile on the server-side. By modifying a user's profile, a usability analyst can choose the events to be captured for a user.

6. References

Byrne, M.D.; John, B.E.; Wehrle, N.E. and Crow, D.C. "The Tangled Web We Wove: A Taxonomy of WWW Use." *Proceedings of CHI'99*, Pittsburgh, PA, (May 1999), ACM Press, (1999), 544-551.

Choo, C.W.; Deltor, B. and Turnbull, D. "A Behavioral Model of Information Seeking on the Web: Preliminary Results of a Study of How Managers and IT Specialists Use the Web." *Proceedings of the 61st ASIS Annual Meeting*, Pittsburg, PA, Cecilia M. Preston (Ed.), Vol. 35, 290-302, Information Today Inc.

Choo, C.W.; Deltor, B. and Turnbull, D. "Information Seeking on the Web: An Integrated Model of Browsing and Searching." *1999 ASIS Annual Meeting*, Washington, DC.

Dix, A.; Finlay, J.; Abowd, G. and Beale, R. *Human Computer Interaction.* Prentice-Hall, (2004).

Drott, C.L. "Using Web Server Logs to Improve Site Design." *Proceedings of ACM SIGDOC 98*, ACM, (1998).

Ellis, R.D.; Jankowski, T.B.; Jasper, J.E. and Tharuvai, B.S. "Listener: A Tool for Client-side Investigation of Hypermedia Navigation Behavior." *Behavior Research Methods, Instruments and Computers*, Vol. 30, No. 6, (1998), 573-582.

Etgen, M. and Cantor, J. "What Does Getting WET (Web Event-logging Tool) Mean for Web Usability?" *Proceedings of the 5th Conference on Human Factors and the Web*, Gaithersburg, Maryland, (June 1999).

Fenstermacher, K. and Ginsburg, M. "Client-side Monitoring for Web Mining." *JASIST*, Vol. 54, No. 7, (2003), 625-637.

Gamboa, H. and Ferreira, V. "Widam: Web Interaction Display and Monitoring." *ICEIS*, No. 4, (2003), 21-27.

Goecks, J. and Shavlik, J. "Learning Users' Interests by Unobtrusively Observing Their Normal Behavior." *Proceedings of the 5th International Conference on Intelligent User Interfaces*, (2000), 129-132.

Gonzalez Rodriguez, M. "ANTS: An Automatic Navigability Testing Tool for Hypermedia." *Proceeding of the Eurographics Multimedia '99 Workshop*, Milan, Italy; *Multimedia '99*. Vienna, Austria, Springer-Verlag, (2000), ISBN: 3-211-83437-0.

Gil, Juan Miguel. "Development of a Tool for the Design and Analysis of Experiments in the Web." *Interaction-04*, Mayo, Spain, (2004).

Hammontree, M.L.; Hendrickson, J.J. and Hensley, B.W. "Integrated Data Capture and Analysis Tools for Research and Testing on Graphical User Interfaces." *Proceedings of CHI'92*, (May 2-7, 1992).

Hartson, Rex H.; Castillo, J.C.; Kelso, J. and Neale, W.C. "Remote Evaluation: The Network as an Extension of the Usability Laboratory." *Proceedings of CHI'96*, (April 13-18, 1996), ACM Press, (1996).

Helms, J.; Neale, D.C.; Isenhour, P.L. and Carroll, J.M. "Data Logging: Higher-level Capturing and Multi-level Abstracting of User Activities." *Proceedings of the 40th Annual Meeting of the Human Factors and Ergonomics Society*, (2000).

Hilbert, David M. and Redmiles, David F. "Agents for Collecting Application Usage Data over the Internet." *Proceedings of Autonomous Agents '98*, (1998).

Hilbert, David M. and Redmiles, David F. "Extracting Usability Information from User Interface Events." *ACM Computing Surveys*, Vol. 32, No. 4, (December 2000), 384-421.

Hong, Jason I. and Landay, James A. "WebQuilt: A Framework for Capturing and Visualizing the Web Experience." *Proceedings of the 10th International World Wide Web Conference (WWW10)*, Hong Kong, (May 2001), 717-724.

Kangas, S. and Chiu, A. "Learning from Event Logging." White paper at NetConversions Inc., June 2001. http://www.netconversions.com/

Laus, F.O. "Tracing User Interactions on World Wide Web Pages." Manuscript 2001, available at: http://wwwpsy.uni-muenster.de/inst3/AEKeil/laus/.

Marques, E.; Garcia, A.C. and Ferraz, I. "RED: A Model to Analysis Web Navigation Patterns." *Proceedings of the Workshop on Behavior-based User Interface Customization*, IUI/CADUI Conference, (January 2004).

Nielson, J. *Usability Engineering.* Academic Press, (1993).

Paganelli, L. and Paterno, F. "Intelligent Analysis of User Interactions with Web Applications." *Proceedings of the 7th Intl. Conf. on Intelligent User Interfaces*, New York, (January 2002), 111-118, ACM Press.

Perkowitz, M. and Etzioni, O. "Towards Adaptive Websites: Conceptual Framework and Case Study." *Artificial Intelligence,* Vol. 118, (2000), 245-275.

Reeder, R.W.; Pirolli, P. and Card, S.K. *WebLogger: A Data Collection Tool for Web-use Studies.* Xerox PARC, Technical Report UIR–2000–06, (2000).

Scholtz, J.; Laskowski, S. and Downey, L. "Developing Usability Tools and Techniques for Designing and Testing Web Sites." *Proceedings of the 4th Conference on Human Factors and the Web*, (1998). See: http://www.research.att.com/conf/hfweb/index.html

Shah, I. and Jawed, M. *Web Event Logging: A Survey.* Research Report No. 3, (2006), Research Center of College of Computer & Information Sciences, King Saud University, Riyadh.

Shahabi, C. and Banaei-Kashani, F. *A Framework for Efficient and Anonymous Web Usage Mining Based on Client-side Tracking.* WEBKDD-2001, Mining Web Log Data Across all Customer Touch Points, New York: Springer-Verlag, (2002), ISBN: 3-5404-3969-2.

Spiliopoulou, M. and Pohle, C. "Data Mining for Measuring and Improving the Success of Websites." *Data Mining and Knowledge Discovery, Special Issue on Electronic Commerce*, (2000).

Spiliopoulou, M. "Web Usage Mining for Website Evaluation." *Communications of the ACM,* Vol. 43, No. 8, (2000), 127-134.

Tiedtke, T.; Martin, C. and Norbert, G. "AWUSA: A Tool for Automated Website Usability Analysis." *9th Intl. Workshop on Design, Specification and Verification of Interactive Systems*, Rostock, Germany, (June 2002).

RWELS: نظام تسجيل أحداث الويب عن بُعد

عناية الله شاه، ول. التعيمي، ومحمد جاويد

قسم علوم الحاسب، كلية علوم الحاسب والمعلومات

جامعة الملك سعود، الرياض، المملكة العربية السعودية

(قدم للنشر في ٩/٩/٢٠٠٦م؛ وقبل للنشر في ٢/٧/٢٠٠٧م)

ملخص البحث. يمثل "سِجِل الأحداث" مصدرًا هامًّا للتعرف على مشاكل الاستخدام في مواقع الويب. نقدم في هذا البحث نظام تسجيل أحداث الويب عن بُعد والذي يعمل بتقنية الويب ويستخدم نموذج الزبون–الخادم لتسجيل أحداث واجهة برنامج التصفح "مايكروسوفت إكسبلورر" أثناء تفاعل المستخدم مع صفحات الويب. يُسجل النظام الأحداث بطريقة تلقائية وبدون مضايقة المستخدم. يتمتع النظام بمرونة عالية ويوفر إمكانية تسجيل الأحداث ذات العلاقة بالمستخدم. محلل النظم يمكنه تحديد صفحات الويب التي يرغب في تسجيل أحداثها واختيار نوعية هذه الأحداث لكل مستخدم. يتم إرسال الأحداث المسجلة عن طريق الويب إلى الجهاز الخادم بحيث تُخزن هناك على شكل ملفات نصية، بحيث يكون كل ملف نصي مرتبطًا بفترة استخدام معينة.