Script-based system for monitoring client-side activity

Artur Kulpa, Jakub Swacha, Roman Budzowski
Uniwersytet Szczeciński
Instytut Informatyki w Zarządzaniu
ul. Mickiewicza 64, 71-101 Szczecin
arturro.q@gmail.com,
jakubs@uoo.univ.szczecin.pl,
romek@innotech.pl

Abstract

In this paper, a system for monitoring client-side activity is described. The system is capable of monitoring a wide range of user actions performed in web browser environment. Event logs are sent to the web server for further processing. The described system is completely script-driven on the client side, requiring no additional software to be installed on the client system and guaranteeing platform-independence. The website user has full control over the monitoring and can disable it anytime.

1. Introduction

Gathering information on customer activity is a well established practice in e-commerce. So-called clickstreams consisting of a list of active elements clicked by website user during his session can be gathered and analyzed server-side, unfolding a pattern in user's preferences [Andersen 2000].

Another notion comes from the field of application usage study [Hilbert 2000]. In this case, any kind of trackable user-triggered event may be recorded, and then processed in order to discover high-level actions, so that application user interface may be rearranged by removing unused capabilities and making frequent actions easier to be performed.

This is also useful in e-commerce, as it is obvious that a well-designed website can generate much more profit than a badly designed one. But the record of user actions at the fine level of detail also constitutes an important source of information for operational customer relationship management. It provides the sales force automation with a very precise description of customer behavior, and the customer service and support with thorough description of circumstances which led to posting an inquiry or a complaint.

However, it requires a client-side monitoring as most of the low-level activity does not have any impact on the server. In today's world of malicious spyware, web users are hardly willing to install any monitoring software.

Thereinafter, we shall present a monitoring system which enables tracking many user actions that may be found useful in the study of improvement of e-commerce web-site. The system does not require any additional software to be installed on the client, as all the tracking code is included in the website itself in the form of client-side scripts. Thus, a user has full knowledge on what is monitored – as the script source code is unhidden – and can easily turn off the monitoring just by changing web browser script-running options (though there is a subtler way of doing that embedded in the system).

2. User activities worth investigating

A client-side monitoring system can track a lot more than a server-side one. Not every user action can be traced even in the client system: eye movements and uttering curses being good examples of untraceable events. However, there is still a colossal amount of data which can be collected. For the sake of its simplicity and resource saving, a monitoring system should limit its surveillance to actions which record may be then found useful in practical purposes.

Such purposes are related either to system usability study or to customer research. Hereunder, some of the most popular purposes are listed:

- feature usage reports: once known, the most frequently used features should be made simpler to use, the unused features should be promoted (in case the users do not know they exist), explained (if users do not know why or how to use them), or removed (in case nobody really needs them); the simple element clicking and keystroke recording is enough to satisfy this purpose
- feature usage pattern discovery: if some features are often used together, they should be available in proximity, if some features are almost always used together, they should be composed into one new feature; in addition to standard click and keystroke events, actions like marking and clipboard operations should also be tracked for this purpose
- usage error reports: these can be helpful in improving user interface by rearranging the size and placement of visible elements, and renaming the feature captions along with adding or improving the help/hint subsystem; the basic actions of importance here are "do then undo" event patterns and also clicking on inactive web page elements
- user history reports: knowing what an individual did in the past may be helpful in personalizing the system for him/her, evaluating the customer's profitability, and – after discovering behavior patterns – grouping the users by some criteria; some sort of identification is obviously necessary here – whether user-oriented (long-term cookies and login data) or session-oriented (temporary cookies)

- browsing reports: these can show which information actually interests the user and which does not; although not a perfect measure, the time of stay on a page could give some clues on that.

3. Monitoring systems nowadays

Tracking user actions for further analysis is almost necessary in order to develop and deploy quality applications, standalone programs and web applications alike. In case of web applications it may have form of client-side or server-side monitoring.

The client-side monitoring of user actions requires either integration of monitored application with monitoring system or installation of additional program capable of recording user actions.

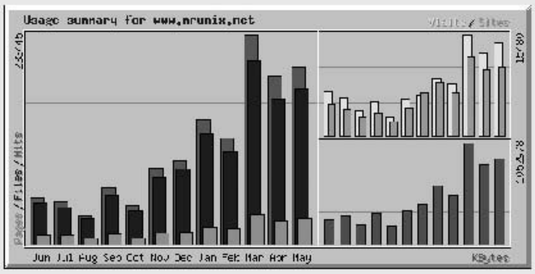
A good example of a client-side monitoring tool is WinTask application from TaskWare [Taskware 2005]. This application is capable of recording mouse movement and keystrokes. Every recorded sequence can be replayed later or used to produce a report. WinTask can also automatically test applications, measuring their performance in form of application response times, critical points and generated network traffic.

Fenstermacher and Ginsburg [Fenstermacher 2002] took a different approach, trying to monitor creating and accessing individual information, not individual application, so that cross-application information flows could be exposed.

Monitoring of web applications is mostly done server-side. An outgoing browser requests leave tracks in server logs. These logs can be then analyzed using data-mining applications such as OpenSource Webalizer [Webalizer 2005] which is a fast and free web server log file analysis program producing highly detailed website usage reports (see Fig. 1 on the next page).

Client-side scripts can be integrated with monitored website to send information about user and viewed page to a third party server, where it is stored in a database. Existing client-side scripts are limited to tracking information which could just as well be tracked using a server-side monitoring. They come practical when a website owner does not have access to server log files or simply wants to present reliable data to others. Such script-based traffic monitoring services are offered, for instance, by gemius.pl [Gemius 2006], stat.pl [Stat.pl 2006], ShinyStat [ShinyStat 2006], OpenTracker [OpenTracker 2006] and many other providers.

Fenstermacher and Ginsburg [Fenstermacher 2003] suggested client-side monitoring framework for web browser activity. Although they described their solution as unobtrusive to users, it is based on a Python program monitoring Microsoft Internet Explorer using Component Object Model (COM) technology, and therefore it requires installing additional software. Moreover, this makes it possible only to monitor actions of clients using Windows operating system and Internet Explorer browser.



Month	Daily Avg Hits	Files	Pages	Visits	Monthly Totals Sites	KBytes	Visits	Pages	Files	Hits
May 1999	6377	5570	933	455	10484	834568	14115	28004	172671	197696
Apr 1999	6216	5394	858	415	10087	821568	12504	25758	161844	186504
Mar 1999	7530	6583	1076	469	12128	1052978	15480	33432	204059	233445
Feb 1999	4712	4138	656	321	6629	511793	8998	18419	103303	117816
Jan 1999	4470	3934	607	284	8079	605694	8808	18844	121980	138571
Dec 1998	2968	2673	411	197	6124	410110	6120	12769	82875	92011
Nov 1998	2910	2567	417	199	4560	347705	5988	11627	74468	84403
Oct 1998	3052	2658	457	202	2203	185253	2805	5299	37260	42738
Sep 1998	2072	1825	345	165	3455	314492	5075	10076	54807	62165
Aug 1998	1014	901	211	123	2693	176560	3800	[illegible]	27938	31455
Jul 1998	1424	1325	302	184	4141	278225	5716	7383	41102	48019
Jun 1998	1707	1503	323	222	7809	351502	6675	9687	45077	51227
Totals						5883848	94952	188269	1127404	1284990

Figure 1. Webalizer – analysis example.

Although existing client-side scripts are limited to traffic monitoring, web browser scripting environments nowadays are complex enough to monitor practically every user action reported by web browser.

4. Sketch of the script-based system

The idea at the core of the script-based monitoring system is to let a web vendor collect a detailed record of a user activity without necessity to install any software on the client system.

Another desired feature is to let the monitoring system work with existing web pages without having to alter them in a complicated way.

The system is a client-server solution, with a client being a web browser capable of running scripts.

The main problem to deal with is obtaining necessary detailed information on occurring events. This can be achieved using the Document Object Model (DOM) [Robie 2004] implemented in most web browsers, although with minor differences to the official standard.

In a DOM-compliant browser, every trackable user action leads to setting up an Event class object whose properties contain information describing the action, like the document element being the object of the action [Pfaffenberger 2004].

Every kind of DOM object available in browser is accessible using JavaScript programming environment [Goodman 2004]. This language can also be used to transmit the gathered event information to the server-side monitoring application. The server application's role is to receive the information from the client, pre-process it and then store it in a database system for further usage.

As for the server-side technology, there are many candidates like PHP, JSP, ASP or any programming language used via Common Gateway Interface (CGI). PHP5 [Coggeshall 2004] has been chosen for the experimental system implementation, as a solution which is effective, free, and popular among host providers.

The website requires a WWW server to be published. The monitoring system requires the server to be able to run server-side scripts. The experimental system implementation uses Apache server [Laurie 2002], as it has huge capabilities and perfectly cooperates with PHP5.

Data can be stored server-side in many ways. They can be saved as they come, arranged into XML documents, or passed to a database system. The latter solution makes the further processing of data much easier. Therefore, the experimental system implementation uses PostgreSQL database system [PostgreSQL 2005], because, once again, it is highly efficient, free, and easily accessible using PHP5.

5. Script-based monitoring in detail

Script-based monitoring software is based on the Event class objects of the Document Object Model (DOM) [Robie 2004]. As many as 25 event types (the actual number may vary between different web browsers) can be registered:

- element-related events (onclick, onchange, onfocusin, onfocusout, onselectstart, ondblclick and oncontextmenu),
- mouse events (onmouseover, onmouseout, onmousedown, onmousewheel and onmouseup),
- drag&drop events (ondrag, ondragenter, ondrop, ondragend, ondragleave and ondragstart),
- keyboard events (onkeydown, onkeyup and onkeypress),
- clipboard events (oncopy, onpaste and oncut),
- script (de-)initialization (onload, onunload).

The event-registering script (*eventlog.js*) is attached to the monitored HTML page via the following inclusion into the HEAD element: <SCRIPT TYPE = 'text/javascript' SRC = '/eventlog.js' LANGUAGE = 'JavaScript'></SCRIPT>.

The *eventlog.js* file must be placed in the root directory of the WWW server. It is possible to automate the process of applying the monitoring script inclusion for the entire website with an appropriate PHP script using regular expressions.

On document loading, the browser attaches the monitoring code to the document's BODY element. It is then activated for every of the aforementioned events. Since then, every time an event is triggered after some user action, the monitoring code is executed. This is possible thanks to the 'bubbles' mechanism, by which in case of an event for any document element, its parent element is notified as well. As the BODY element is the (grand-)parent of all the elements in the document, no event can evade being registered.

The proposed system does not interfere in any way with existing scripts (other tracking system or anything else), as the event handlers are not replaced - they are only wrapped with the monitoring code. Therefore, there is no need to alter the existing scripts in any way.

The information being registered by the monitoring system depends on the kind of the event. Some parts of it are common for any kind of event:

- the document's URL
- client-side date and time of the event
- kind of the event
- kind of the element which triggered the event
- NAME attribute value (provided it exists).

The remaining are either element type dependent:

- for INPUT elements: their type and value
- for A elements: their reference (HREF)
- for FORM elements: their action
- for other elements: first 20 characters of their content;

or event type dependent:

- for clipboard events: selection and clipboard contents
- for drag&drop events: selection contents
- for keyboard events: the keystroke
- for mouse events: mouse cursor coordinates.

The registered information is stored as text in the following format: [information_type == content;;].

To provide user identification capability, a user/session registration mechanism has been built into the monitoring system. When a user enters the monitored website for the first time, two identification cookies are created. The first cookie for the purpose of session identification is temporary, valid only until the user leaves the website. The second cookie for the purpose of user identification has validity period 30 days long, thus it allows to identify users returning to the monitored website after a short period of time, but no longer than one month. The values of these cookies are attached to every recorded event information.

In case the client browser cleans cookies after leaving the website, it will not be possible to identify the user returning to the site unless the website requires a

login. In case the client browser has the cookies completely turned off, it will not be possible to continue the monitoring session when the user switches to another page on the website, limiting the session to one web page. Notice that even in this case the entire website monitoring session can usually be reconstructed later from server-side logs containing user's IP record.

The gathered information is sent to the monitoring server by setting up an asynchronous request. In case of Microsoft Internet Explorer environment 'Msxml2.XMLHTTP' ActiveX object is used for this task, whereas in Mozilla environment - 'XMLHttpRequest' object. Upon receipt of such request, the server executes a recording script written in PHP language. This script attaches the server-side date and time to the received event information and then stores it all in a PostgreSQL database system.

Communicating with the monitoring server every time an event has been triggered could noticeably slow down the browser and produce too much web traffic. In order to overcome that, the event records are merged into packets. The packet is only sent to the server when the number of stored events exceeds 20 or on the document unload event (leaving the site by the user).

All the gathered data are sent to the monitoring server using the same protocol as the remaining communication. Therefore, if the connection is secured, then the monitoring reports are sent in encrypted format. There is no possibility of sending the data from the client to anywhere outside the monitoring server domain. The gathered data are as safe as the web server is.

For additional security, it is possible to turn off monitoring specific kinds of events (e.g. clipboard operations) or gathering data on specific kinds of controls (e.g. input password boxes). The user can also disable the monitoring for a single web page or the entire website.

6. Test results

The described monitoring system was tested on two websites: www.klubinformatyka.pl and www.konferencja.org (Fig. 2). The former is a content delivery site, therefore user activity there is rather simple and concentrated on a website navigation. The latter is the biggest Polish scientific conference catalogue and search engine, allowing user to specify search criteria, insert conferences into the catalogue and add a chosen conference to "My conferences" private repository. User activity here is much more complex with more types of actions being performed.

The monitoring scripts were installed on both websites with no special adaptation.

During one-week survey the script-based monitoring system proved its suitability in user activity tracking by managing to produce excellent detailed event reports. Although the widest range of events was reported from Internet Explorer, most important actions were also captured on FireFox (and all

browsers based on Gecko engine), though lacking reporting of, for example, clipboard-related events.

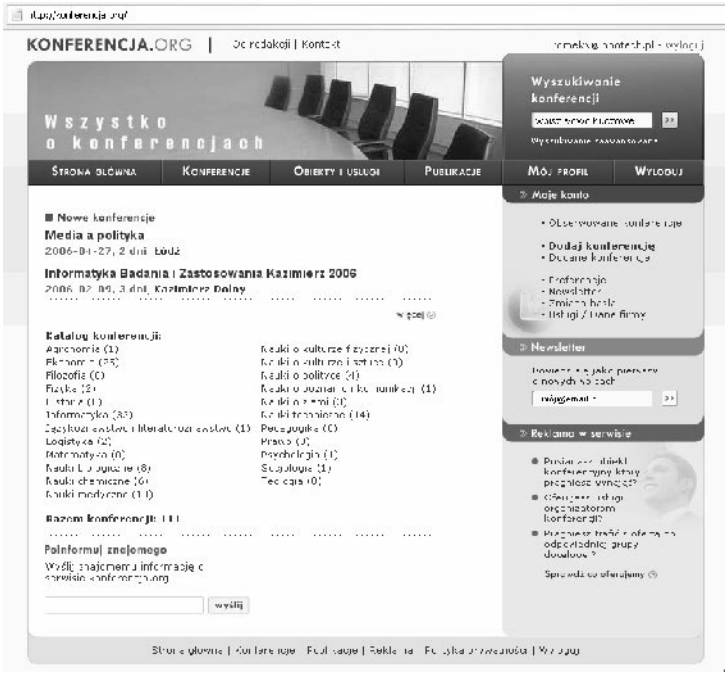


Figure 2. The www.konferencja.org main page.

The easy script configuration process with the capability of turning off or on each type of reported events separately makes it possible to focus the survey on selected features of the website.

In order to limit the impact of monitoring on the monitored system performance and generated web traffic, the packet size should be small.

After one week of monitoring, gathered data allowed to estimate average event description length and packet size (see Table 1). As the former amounted to 385 bytes, 20 consecutive events resulted in a packet of 7.5 KB. This is not much for high-speed Internet connections, and it can also be greatly diminished by applying a fast data compression algorithm (e.g. [Swacha 2004]), which was not implemented in the experimental system only for the sake of its simplicity.

Table 1. Measured packet and event description size.

Data	Value
Registered events (one week)	43661
Average packet size	7700 B
Average event description length	385 B
Maximum event description length	440 B
Minimum event description length	213 B

The time gap between sending consecutive packets (see Table 2; very long inactivity periods are not included as they break the sessions) depends on the type of the monitored website. It gets longer for websites with more content, keeping users busy reading, thus reducing their activity.

Table 2. Time gap between packets uploads .

Data	Time (s)
Average time	3
Minimum time	1
Maximum time	15

Moving the mouse across the screen over many active elements produces a packet every second. Such a frequent packet uploading may noticeably slow down the Internet connection, bringing up the compression issue again. Diminishing the packet size is necessary for the proposed monitoring system to be used unobtrusively even on slow dial-up Internet connections.

7. Conclusions

The presented system for monitoring a client-side activity is capable of monitoring a wide range of user actions performed in a web browser environment. The actions are recorded, arranged in packets, and then sent to the server to be stored and processed.

Stored information contain detailed facts about the element which triggered the event, precise date and time of the event, and identification of the document, user and session during which the event took place. Although discussing how this information can be processed is beyond the scope of this paper, it is obvious that it can at least yield valuable information on usability of particular web page elements and errors in their arrangement, thus giving hints to improve the quality of the website.

A unique feature of the described system is that it is completely script-driven on a client side, requiring no additional software to be installed on the client system. It also makes the system platform-independent as well.

8. References

[Andersen 2000] J. Andersen, A. Giversen, A. H. Jensen, R. S. Larsen, T. B. Pedersen, and J. Skyt, *Analyzing Clickstreams Using Subsessions*. In *Proceedings of the Third International Workshop on Data Warehousing and OLAP*, Washington DC, November 2000, pp. 25–32.

[Coggeshall 2004] J. Coggeshall, *PHP Unleashed*, Sams Publishing, Indianapolis, 2004

[Fenstermacher 2002] K.D. Fenstermacher, M. Ginsburg, "A Lightweight Framework for Cross-Application User Monitoring", *Computer*, 35(3), 2002, pp. 51–59

[Fenstermacher 2003] K. Fenstermacher, M. Ginsburg, "Client-Side Monitoring for Web Mining", *Journal of the American Society of Information Science and Technology*, 54(7), 2003, pp. 625–637

[Gemius 2006] *Gemius*, www.gemius.pl, 2006

[Goodman 2004] D. Goodman, M. Morrison, *JavaScript Bible*, Wiley Publishing, Indianapolis, 2004

[Hilbert 2000] D. Hilbert, D.F. Redmiles, "Extracting Usability Information from User Interface Events", *ACM Computing Surveys*, Dec. 2000, pp. 384–421

[Laurie 2002] B. Laurie, P. Laurie, *Apache: The Definitive Guide*, O'Reilly Media, 2002

[OpenTracker 2006], *OpenTracker*, www.opentracker.net, 2005

[Pfaffenberger 2004] B. Pfaffenberger, S.M. Schafer, Ch. White, B. Karow, *HTML, XHTML and CSS Bible*, Wiley Publishing, Indianapolis, 2004

[PostgreSQL 2005] *PostgreSQL*, www.postgresql.org, 2005

[Robie 2004] J. Robie, S. Byrne, P. Le Hégaret, A. Le Hors, L. Wood, G. Nicol, M. Champion, *Document Object Model (DOM) Level 3 Core Specification*, http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-200404077, April 2004

[ShinyStat 2006], *ShinyStat*, www.shinystat.com, 2005

[Stat.pl 2006] *Stat.pl*, www.stat.pl, 2006

[Swacha 2004] J. Swacha, *The Predictive-Substitutional Compression Scheme and its Effective Implementation*. In *Proceedings of Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, 2004

[Taskware 2005], *Wintask*, www.wintask.com, 2005

[Webalizer 2005] *The Webalizer*, www.webalizer.com, 2005