CHITAS - a mouse tracking system in a web environment

A Jevremovic, Univerzitet Sinergija, N. Ristic, Univerzitet Sinergija, and M. Veinovic, Univerzitet Singidunum

***Abstract*— Eye tracking may be very effective and has great potential in user interface optimisation. The main disadvantage of tracking what a user is looking at is the need for complex and expensive equipment, and a controlled laboratory environment. Mouse tracking is a widely applicable alternative to eye tracking. This paper presents the architecture of CHITAS (Computer-Human Interaction Tracking and Analytics System) which enables information to be gathered about mouse movement in a web environment, and analytical processing of the collected data.**

Keywords – ***Eye tracking; mouse tracking; PHP; visualisation***

I. INTRODUCTION

Eye tracking may be very effective and has great potential in user interface optimisation. One of most common uses of the information gathered using eye tracking is in the optimisation of the user interface [1] [2].

The main disadvantage of research using eye tracking is the need for complex and expensive equipment and a laboratory controlled environment, which significantly narrows the field of application for this system. In addition to the complex hardware used for tracking the user's pupil, there is complex software running in the background of the eye tracking system – working by finding pupils on a saved image [3], to matching that image with the position of the screen. The most widely applicable alternative to eye tracking is mouse tracking. The main reasons for using mouse tracking are the low tech demands, its simplicity and a high rate of correlation between approaches [4]. This method can be used on the internet, for example, to track user interactions with different web applications [5].

This paper presents the architecture of the CHITAS system (Computer-Human Interaction Tracking and Analytics System) which enables the collection of information about mouse tracking in a web environment, and also the analytical processing of the collected data.

II. TRACKING SYSTEM ARCHITECTURE

When evaluating website design, commonly used methods such as A/B testing, are relatively easy to implement but generally give only very general results. A significant failure of A/B testing, especially when it comes to sites with relatively low traffic, is the fact that it takes a long time to obtain statistically verified conclusions. For example, on a site that is visited daily by approximately 200 visitors, with an efficiency of 20%, confirmation of a 10% increase in efficiency would require 64 days of testing. This calculation refers to the testing of only two variants, the site or any part thereof. In the case of A/B using the ten tested embodiment, the time needed to produce statistically significant results is 320 days. In most real situations the specified times are not acceptable because the results could be obsolete by the time they are produced [6].

For visitor tracking purposes we developed a system called the Computer-Human Interaction Tracking and Analytics System - CHITAS. It's a client-server solution that is easily integrated into the web page we want to track by adding a JavaScript code snippet. Tracking is initialised after the "document ready" page loading event is fired. The JavaScript component involves collecting data about various aspects of user behaviour (mouse movements, clicks, text selections, keys pressed...) and sending it as JSON encoded data to a server via AJAX calls. The component on the server side is a web application where PHP is accepting data from the client side and is storing it to a MySQL database.

Another part of the CHITA system is dealing with the visualisation/analytic of collected data. Some functions, mostly for data filtering and exporting, are realised directly in PHP, however, for greater flexibility, visualisation and more complex analyses are done on CVS exported data within tools like R or MatLab.

The mouse tracking system we developed for our research is able to detect five levels of objects that are involved in presenting web content: screens, windows, viewports, pages and wrappers. The screen is area of physical display, a matrix of pixels with a dynamic width and height.

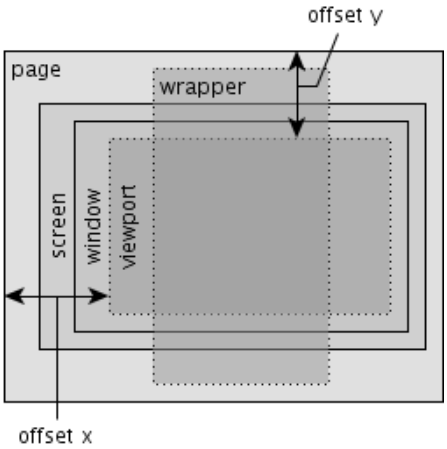

Figure 1. Display structure

DOI: 10.7251/ZRSNG1501131J

The window area is the web browser's GUI window object, with variable width, height and position on screen. Pixels in all windows are usually displayed within the screen area and the system has detected that the most of visitors are using Web browser within maximized window. The system is also able to detect whether a window is active (in focus) or not (minimised, or behind some other window) which is also important for our research.

The viewport area is a “useful” part of a web browser's window, or the part where a loaded web page is actually displayed. The size of this area is equal to the window's size minus the window title and border, toolbars and status bar. Scrollbars, if any, are included within the viewport area. When the web browser is in full-screen mode, the viewport area is equal to the window size and screen size.

Page area is a representation of the body element within a HTML document. The page area can fit within the viewport area if the page's content doesn't require more space for display. Where that content's display size is less than the available viewport size, the page area is expanded to fit the viewport size. Conversely, if displaying content requires more space than viewport area, page size is adapted to content size and viewport scrollbar(s) are enabled.

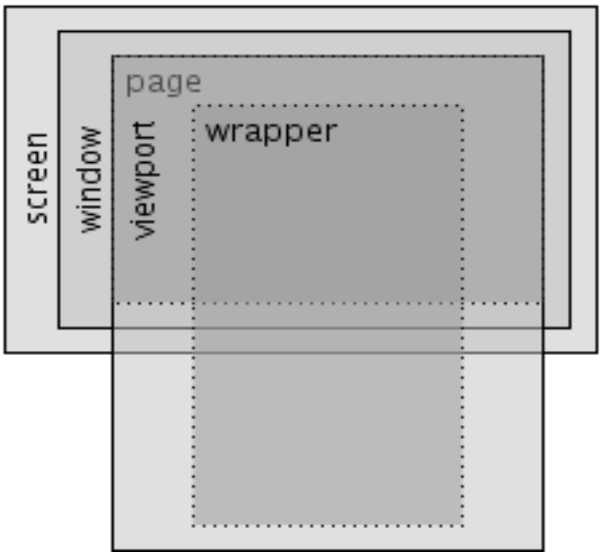

Figure 2. Page with no horizontal or variable vertical offset.

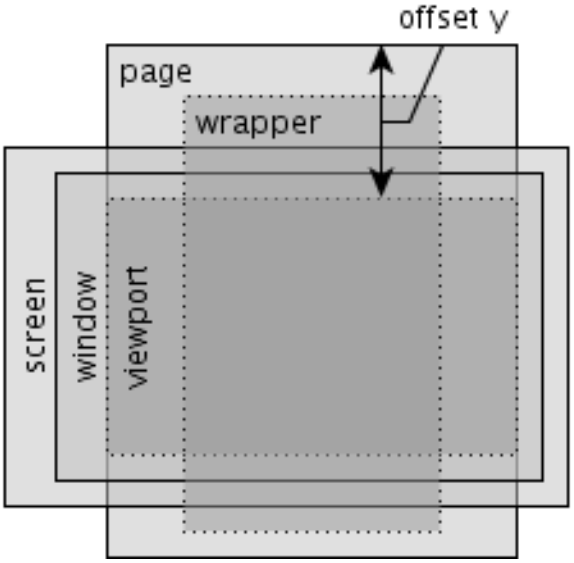

Figure 3. Page with no horizontal or variable vertical offset.

Offsets are a measure of how many pixels a page scrolls vertically or horizontally. Even if we are able to directly calculate the mouse position relative to the page/wrapper, putting it in a viewport/offset context can provide additional useful data, because users interact differently with different viewport parts [6].

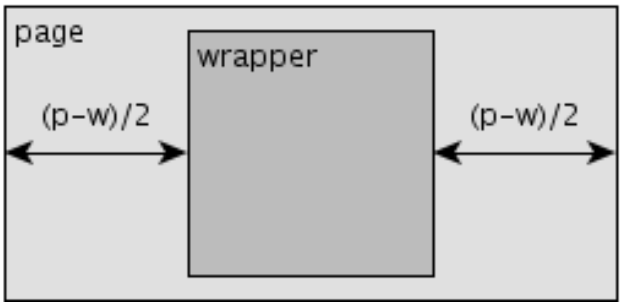

Figure 4. Wrapper position with “auto” left and right margins.

the wrapper is the page content part on which our tracking is focused. It can be any displayed part of the content, but is usually the DOM element within the body element where whole page content is contained. In our case, we used one div element with the “TPL_Wrapper” value of the “ID” attribute, which was the only child of the body element and which contained all the page content within itself. The wrapper size in our experiment was 878 pixels and both left and right margins were set to “auto” which resulted in a horizontally cantered wrapper area. We consider this (horizontally cantered content) as generally good practice, but especially useful in eye/mouse tracking experiments, because content is displayed directly in front of visitors.

The CHITA System can be implemented as an individual process (tracking server) or as a proxy.

DOI: 10.7251/ZRSNG1501131J

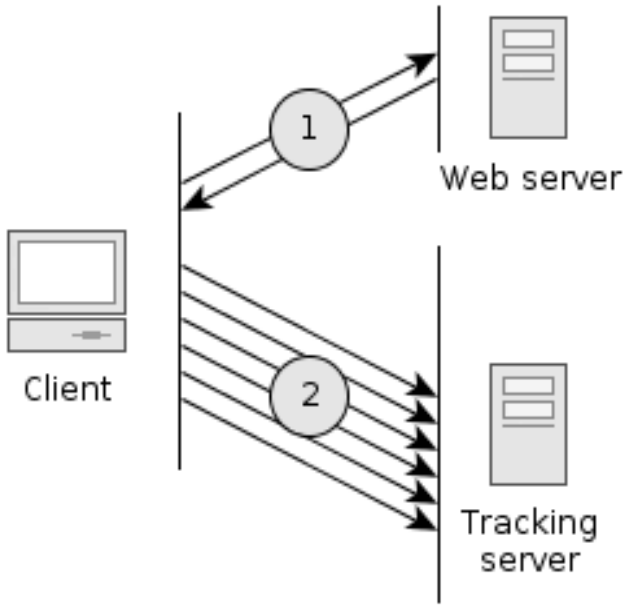

Figure 5. Tracking server model as a standalone service

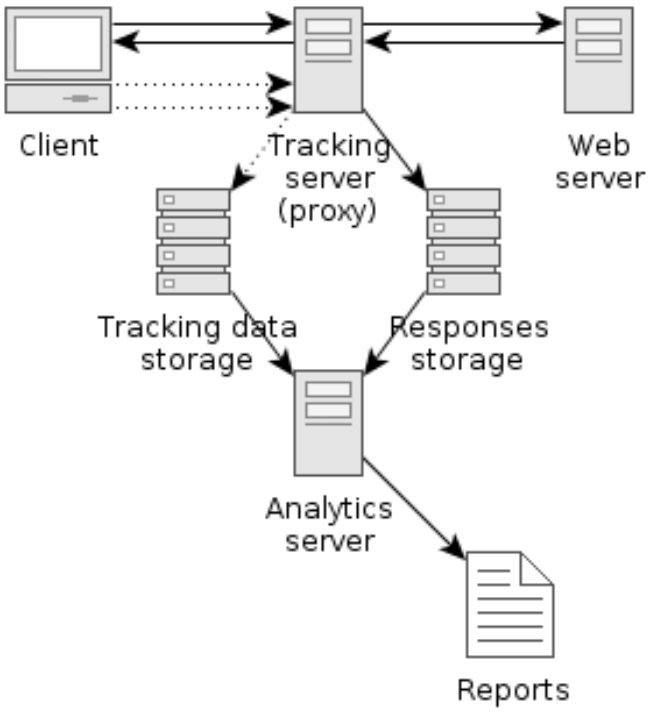

Figure 6. Proxy implementation model

The proxy implementation model is better for implementation when the content of a web application can't be changed and when it is necessary to save a response for later assessment (Dynamic content).

The mouse tracking component on the client side is written in JavaScript language and uses DOM events. Different events are used to provide the necessary mouse tracking data: mousemove, mousedown, mouseup, click, dblclick, mouseover, mouseout, mouseenter, mouseleave, and wheel.

This client-side component is communicating with the server in two ways. Initially, some parameters (unique request ID, visitor's ID, visit's ID...) are inserted in the JavaScript code, during the response generating process.

After the component is loaded and initialised on the client's browser, it periodically sends captured data to the server-side component within AJAX requests. This synchronisation is done every second, no matter whether there are captured events or not.

A. Protocol for mouse tracking data collecting

The communication protocol between client-side and server-side components of our system is based on AJAX (XMLHttpRequest) requests and uses HTTP(S) as the underlying protocol. All requests are JSON encoded matrices containing captured events parameters. In addition to regular DOM events, two more event types are added: register and ping.

A register event is fired and sent to the server immediately after a page is loaded. Within this request to the server some general parameters are sent - requested hostname and URL, visitor's IP and UserAgent attributes. Because of the large size of this data, it is transferred to a server only once, within the first, register event.

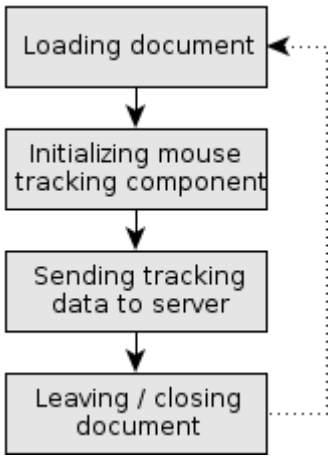

Figure 7. Possible states of client-side protocol

A ping event is used for synchronisations, when no events are captured until the last synchronisation. We found this event useful for monitoring whether a web page is still open client side, and for keeping connections alive. There is also a synchronisation counter on both sides (similar to a TCP window size mechanism) which prevents lost data. We performed some experiments with request compression, in order to decrease request size, but the ratio between saved bandwidth and data loss has not justified inclusion of this functionality.

B. Database and bandwidth considerations

The database table that contains captured events parameters has 40 columns, and the average record size is about 300 bytes. For the purpose of this paper we saved 3,310,894 events to the database, which required around 1GB or space ~300MB table for 3,000,000 inputs. For 560 visits,

DOI: 10.7251/ZRSNG1501131J

3500 pages, and around 5 minutes averagely per minute for visit:

- ~857 lines per opened page -> 2.86 lines per second (even distribution)
- 2.86 x 300B = ~1KB/s per user

These calculations should be considered in the planning infrastructure for implementing tracking systems on more intensively used web sites. For example, we tested tracking system robustness on one website that at its peak is used by nearly a thousand parallel visitors. The result was around 1MB/s of incoming tracking data that should be received by the server and stored in a database. A database of that size, with that level of use, would reach 1TB in less than two weeks.

III. CONCLUSION

This paper presents the architecture of system CHITAS (Computer-Human Interaction Tracking and Analytics System), which enables information about mouse tracking in a web environment to be collected, and also the analytical processing of collected data.

The Computer-Human Interaction Tracking and Analytics System solution is easily integrated into web pages by adding a JavaScript code snippet. Tracking is initialised after the “document ready” page loading event is fired. The JavaScript component collects data about various aspects of user behaviour (mouse movements, clicks, text selections, keys pressed...) and sends it out as JSON encoded data to servers via AJAX calls. The component on the server side is a web application where PHP accepts data from the client side and stores it to a MySQL database.

Another part of the CHITA System involves dealing with the visualisation/analysis of collected data. Some functions, mostly for data filtering and exporting, are realized directly in PHP, however, for greater flexibility, visualisation and more complex analysis are done on CVS exported data within tools like R or MatLab.

The mouse tracking system developed for our research in this paper is able to detect five levels of objects that are involved in presenting web content: screens, windows, viewports, pages and wrappers. Screen is the area of physical display, a matrix of pixels with dynamic width and height.

REFERENCES

[1] J. Nielsen and K. Pernice, Eyetracking web usability, New Riders, 2010.

[2] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott and A. M. Wichansky, "Eye tracking in web search tasks: design implications”, Proceedings of the 2002 symposium on eye tracking research & applications, pp. 51-58, 2002.

[3] M. Stojmenovic, A. Jevremovic and A. Nayak, "Fast iris detection via shape based circularity”, Industrial Electronics and Applications (ICIEA), pp. 747-752, 2013.

[4] J. Huang, R. White and G. Buscher, "User see, user point: gaze and cursor alignment in web search”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1341-1350, 2012.

[5] A. Jevremovic, M. Sarac and M. V. Milan Milosavljevic, "Analyzing the behavior of students during electronic testing,”, in 1st International Conference on Electrical, Electronic and Computing Engineering, 2014.

[6] A. Jevremovic, S. Ž. Adamović and M. Veinović, "Mousetracking visitors to evaluate efficacy of web site design”, SERBIAN JOURNAL OF ELECTRICAL ENGINEERING, 2014.

DOI: 10.7251/ZRSNG1501131J