A Framework for Collecting Client Side Paradata in Web Applications

Natheer Khasawneh
Rami Al-Salman
Ahmad T. Al-Hammouri
Stefan Conrad
A FRAMEWORK FOR COLLECTING CLIENTSIDE PARADATA IN WEB APPLICATIONS


*College of Computer and Information Technology, Jordan University of Science and Technology, Irbid 22110, Jordan
{hammouri, natheer}@just.edu.jo, ramialsalman9@gmail.com

**Institute of Computer Science, Heinrich Heine University, Dusseldorf, Germany
conrad@cs.uni-duesseldorf.de

ABSTRACT
User behavior on web applications holds valuable information that can be used by web engineers to enhance user experience. So far, most frameworks in tracking user behaviors rely on data solely collected on the server side. Server-side data usually holds limited information, such as IP address, requested URL, and time and date of the request. Client behavior on a page itself, such as mouse clicks, mouse hovering, and text editing, are not recorded on the server side. In this paper, we present a framework that collects and stores clients’ behavior on web applications. The framework is implemented in JavaScript, PHP, and MySQL. The data collected using the proposed framework can be then used in enhancing future user experience with the web application, and in web personalization.

Keywords: Web Engineering, Client-Side Data, Web Personalization, JavaScript.

1. INTRODUCTION
In the recent years, the number of websites has grown vastly. Unlike those websites that were spread during the 1990s—which used to present the information to users with static web pages—current web sites—which we call web applications—interact with users through rich and dynamic contents. Such web applications are complex and contain many interaction items, such as multimedia and JavaScript content. This type of web applications is commonly referred to by Web 2.0. The scripting language JavaScript became the de facto standard for web applications. In the past, it was mainly used for making the websites more interactive with the clients. Lately, JavaScript is employed to do effective tasks, such as decreasing the manipulation tasks assigned to the server side. For example, JavaScript can be used to validate form user-input values. Very recently, JavaScript has evolved to be more interactive not only with the client side but also with the server side. Thus, Asynchronous JavaScript and XML (AJAX) is introduced. The main idea of the AJAX is to allow the JavaScript code to communicate back with the server. Using the XMLHttpRequest object, JavaScript can request data from the server. This allows partially loading parts of a webpage instead of loading the whole webpage at once.

Understanding and evaluating users’ behavior with these applications will help in improving these applications to better serve clients. In the past, the main source of users’ behavior used is server log data. Server logs are restricted to have fixed number of data entries (e.g., IP-Address, Date, etc…). So the process to extract users’ behavior is constrained by the limitations of these logs. Whereas, in the case of client logs data, the number of data entries are not fixed and not uniform and some entries can be extracted from the client’s mouse movements over the visited Webpage. Additionally, the client’s logs data can be formulated to fit exactly into the behavior analysis method that makes the analysis process easier and efficient. Building a complete framework, which can track the client’s data in an efficient way and can extract a useful knowledge from these collected data, will definitely help in understanding clients’ behaviors and attitudes in better way. Web developers and designers can therefore utilize this information for improving the usability and accessibility of the web applications.

2. RELATED WORK
Several client-side data loggers and events capturing systems have been proposed. One of the earliest automatic systems for capturing client-side events is WebVIP [1]. WebVIP is mainly used for tracking the number of different event types. The client’s events are captured and stored in a database “as is” without any preprocessing steps.

Tracking client’s movements and actions is presented in [2], where the authors proposed a JavaScript framework that can handle the client’s events in websites. In addition, these clients’ events are combined with server logs to give more information about clients’ behavior. Finally, based on the collected data, the Webpage was modified by injecting adequate JavaScript code via a proxy server. The authors did not mention how and based on what they modified the Webpage. An approach for detecting client’s browsing patterns is presented in [3]. The detection is performed based on the client’s data. Clients events are presented as a tree that contains each accessed element and its associated accessed time. Then, the Sequence Alignment Method (SAM) is applied to discover the client’s patterns from the navigated elements. The problem is that SAM is applied directly without a
preprocessing step. Furthermore, it is not clear how the events are reordered.

A framework that catches the client’s events and the associated AJAX events is proposed by [4]. The key point was the ability of the framework to catch the executed AJAX events. In addition, the users’ willingness to take part in a remote usability test of a website is evaluated.

Two different software packages for both client’s data and eye tracking are presented in [5]. The first one is called the WebLogger, which records client’s data and used to track eye client’s data. These two types of data are merged and entered as input to the analysis program, which is called WebEyeMapper.

A mouse tracking system using client-side scripting is presented in [6]. The system tracks the mouse movements (position and associated timestamps). The data is then sent to a backend server via the AJAX framework. Additionally, the client’s movements are visualized to understand the client’s behavior. Unfortunately the collected events are not sufficient to understand the client’s behavior.

A Simple Mouse Tracking (SMT) system is proposed [7]. The main idea is to track the client’s mouse movements and to allow the system administrator to visualize these movements.

Clients’ cookies are used as a source of data in the WET [8] JavaScript tool. WET stores and analyzes the client’s cookies that contain some of the client’s preferences related to a given website. The problem in this tool is that the client must explicitly select his favorite cookie option. Furthermore, each website must provide options to transfer and gather user’s data.

3. A FRAMEWORK FOR COLLECTING CLIENT-SIDE DATA

In this section, we present the overall architecture of our proposed system. Fig. 1 shows the four phases of the proposed framework: session identification, events identification and catching, events storing, and merging and exporting events. In the rest of this section, we elaborate on each phase.

3.1 SESSION IDENTIFICATION

During this phase, each period of activity for every unique user and the web server is identified. SessionId, which is a unique number among all identified sessions, is created. SessionId is a function of the client’s IP address and the current time stamp on the local computer, i.e., the server. This guarantees the uniqueness of the sessionId across sessions.

3.2 EVENTS IDENTIFICATION AND CATCHING

During this stage, we identify the web elements and the associated events. The events are classified into two categories: Clickstream based and Time based. Clickstream-based events track events which are related to the clickable web elements. For example, a user could click on either a button or a division (DIV) web element. Time based events tracks the events which are related to the time spent over a web element. For example, a user could hover his mouse over a web element for a given time and then moves away from over that element. So, the time difference between the entrance and the exit of that web element is recorded.

3.3 STORING EVENTS

During this phase, the identified events are stored in relational database on the server side. The data are sent in raw format and are then parsed and stored in a structured format in the database.

3.4 MERGING AND EXPORTING EVENTS

In this phase, events records are grouped per client session (user id) and are then exported as two types of data: General mode and Time based mode.

4. IMPLEMENTATION DETAILS

In this section, we present the implementation details of the proposed framework with excerpts of JavaScript code the database design.

![Figure 1: System Architecture](image)

**Session Identification**

**Events Identification and Catching**

**Events Storing**

**Merging and Exporting Events**

4.1 SESSION IDENTIFICATION

Once a client requests a webpage, the session identification function is fired by the OnLoad event. As shown in Fig. 2, when the Major_In() function is called, a new Date object is instantiated. Then, the session starting time (maj_in variable) value is obtained via the getTime() function. The user_id() function is called which to obtain the sessionId value from the Date object using the getTime() function too. The returned value of getTime() represents the number of millisecond since midnight Jan 1, 1970. By this way, the assigned sessionId for each client is a unique. The generated sessionId is used to identify all recorded events which belong to the same user. Finally, for finishing the current sessionId value and (or) instantiate new sessionId value, the Major_out() function is called. Major_out() function calculates the session end time (maj_out variable), and then the time difference between the start and the end time...
4.2 EVENTS IDENTIFICATION AND CATCHING

To track clickstream based events, once a client clicks on the element, the function inside OnClick event is fired, and the target data is transferred associated with userID via XMLHttpRequest AJAX call (i.e., `<a hrefname="LinkNews" OnClick = "Transfe('LinkNewsValue')">LinkNews</a>`). The transferring data is a lightweight operation, the reason why we used AJAX technology. As shown in Figure 3, the content variable which will be transferred consists of seven variables, which are concatenated and delimited by ",". The seven variables are name, value, Item_time, sessionId, Date, TotalMouses and Personlized. The name variable represents the name of the web element. The value variable represents the value of web element. The Item_time variable, represents the amount of time spent over a specific web element. In other words, Item_time variable stores the time difference between OnMouseOver and OnMouseOut events. The sessionId variable, is a very important variable, and represents a current session id for a specific client. In addition the final sessionId state value will be identified, when the value of personlized variable is not null. The Date variable represents the event date and time. We register the date on the client side for two reasons: to determine the date of the element event and to determine whether the date on the client machine matches the server machine’s date. The TotalMouses variable is a counter which is incremented on every click. Moreover, the final counter value is sent when the Personlized value is not null. The Personlized value indicates if the client session is finished or not. In this framework, the focus is not only on the name and the value of the element but also on the time which is spent on each element. Thus, we inject two types of events: OnMouseOver and OnMouseOut. Once the client sets his mouse over a web element, a function is invoked to instantiate the client’s time object. As shown in Figure 2. This operation can be accomplished using JavaScript’s Date Object. Likewise, when the client moves the mouse away from over the web element, another function, related to OnMouseOut event, is invoked to calculate the time difference between OnMouseOver and OnMouseOut events and to store the value in the Item_time variable. Next, The OnMouseOut event uses the XMLHttpRequest protocol to send the content variable to the server. All concatenated variables inside the content variable are identical to the Clickstream based approach, except that for the newly added variable, Item_time.

4.3 STORING EVENTS
The client data, which is sent via the GET method inside XMLHttpRequest, is received by the backend server (script_page.php). As shown in Figure 4, because the data was concatenated and delimited with ";" on the client side, it must be parsed to extracting the variables values. The explode function in line 2, is used to convert the data into array of variables. In addition, in lines 6 and 7, other variables are initialized at the server side. These are the IP-Address and the server date. In line 8, the variables will be inserted into items_events table. Finally, in line 11, the user id and personalized variables are inserted into the user_select.

Figure 5 shows the Database schema, which consists of two tables, Events table and session Primary key table. The Events table stores vectors of values for the client transaction (events). On other hand, the Session primary key table stores the session id for each client, which is a primary key, and the final personalized item. The relationship between these tables allows us to easily merge data for client sessions.

4.4 MERGING AND EXPORTING EVENTS

The operation of exporting data has two modes: General mode and Time based mode. In a general mode, the exporting is done by a simple aggregation of all records within a specific session. The output of the aggregation operation is a vector data structure consisting of six fields: all clickstreams, the date for each click stream, the session ID, the total spent time in the session, the total number of clicks, and the IP address. Figure 6 shows the snapshot of the aggregated records without both the IP address and the date columns in the general mode.

In the time based mode, the scenario is similar to the general mode, except that while the records are aggregated, the time spent over each item is accumulated. In this case, the vector data structure consists of n fields to store n-accumulated times for each web element in addition to the fields in the general mode but without the clickstream field. Figure 7 shows the snapshot of the aggregated records without both the IP address and the date columns in the time based mode.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a new framework to collect client side Paradata. The proposed framework was implemented using JavaScript, PHP, and MySQL. The
events on the client-side were collected and sent back to the server. At the server side, the data were structured in a convenient way for further applications, such as enhancing user experience, web personalization, and better evaluating user behavior. As a future work, we will apply the proposed framework to different web applications and then apply different data mining techniques to cluster or classify the users of the web application.

ACKNOWLEDGEMENT:
This research was supported in part by the German Research Foundation (DFG), the Higher Council for Science and Technology (HCST) and the Jordan University of Science and Technology (JUST).

REFERENCES: