IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 7, 2020, accepted February 26, 2020, date of publication March 2, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977879

UXJs: Tracking and Analyzing Web Usage Information With a Javascript Oriented Approach

JAIME SOLÍS-MARTÍNEZ[1], JORDAN PASCUAL ESPADA[1], RUBÉN GONZÁLEZ CRESPO[2], B. CRISTINA PELAYO G-BUSTELO[1], AND JUAN MANUEL CUEVA LOVELLE[1]

[1]Computer Science Department, University of Oviedo, 33005 Oviedo, Spain
[2]Computer Science and Technology Department, International University of La Rioja, 26006 Logroño, Spain

Corresponding author: Jaime Solís-Martínez (solisjaime@uniovi.es)

ABSTRACT Knowing what the user does inside your web and how he does it is crucial nowadays to understand the strengths and inconveniences of your web's design and architectural structure as well as about the usability of the site. Currently, there are several solutions that allow the tracking of the user behavior but these have some limitations due to the information they are able to capture and how they can present that information in a useful way for the web developer. Many of these platforms don't capture information about the user activity in the websites, clicks, mouse movements, etc. Some solutions do capture some of this user activity, but they only process the information visually showing heatmaps. In this paper we present UXJs, a novel research approach for collecting automatically all possible information about the user activity in websites, showing this information quantitatively and allowing its automatic statistical analysis and the rapid understanding by web developers.

INDEX TERMS Web, javascript, usability, tracking, statistics.

I. INTRODUCTION

Nowadays, web pages and web applications tend to offer their users a combination between functionality and usability, with this last factor being a key aspect of web environments in combination with accessibility [4]. Web usability is a factor that indicates how easy a user can navigate and make use of a given website and with its increasing importance there are studies with ideas on how to improve the usability of a web page. These ideas include contents regarding how to improve aspects like browsing and user experience [20] and are classified in greater groups of elements like heuristics, recommendations or guidelines [21]. When grouped in heuristics, ideas are considered as a set of steps that allow developers to design systems or elements that are easy to use; Jakob Nielsen published, in 1994, one of the most recognized heuristics collection for user interface design [22]. As useful as these heuristics collections can be, some UX experts state that they are too theoretical to be applied directly to web development [22], so the best form to apply them in this type of scenario is in the recommendations form, as developers can create standards for them. Usability and accessibility have become the key to success for the majority of websites, and a good balance between them is mandatory in order for a website to succeed with its objectives, regardless of the existence of special needs on the user side due to disabilities [5] [6]. Moreover, with the fierce competence there is in the online business, users will choose one alternative or the other based not only on the functionality it offers but also on the ease of use each of the systems provide. That is why currently many web pages change their design and layout frequently: in order to match user demands and based on the data their platforms collect when users navigate through them.

With such situation, there are several approaches intended for the measurement and collection of information regarding user interaction inside a web page, like Google Analytics, Open Web Analytics and Matomo, for example. The collected information can then be studied and analyzed by the web's team in order to identify potential usability and functional problems. However, most of these tools require the modification of the web's source code to some extent in order for the user's interactions to be measured and stored correctly, with these changes adding additional complexity to the interaction monitoring setup process. Thus, our objective

The associate editor coordinating the review of this manuscript and approving it for publication was Dominik Strzalka.

is to create a software component capable of automatically collecting quantitative information regarding the interactions users make in web pages without the need for any configuration on the server side of the system and the code of the website. The collected information includes but is not limited to: mouse movement around the page, mouse clicks registered by each of the elements on the page and total time spent in each of the pages of the web. With this type of information, the administrators of the analyzed website will be capable of understanding the strengths and needs of each of the pages in the site in terms of usability and will be able of establishing the need for modifications in those cases where the figures indicate the need for action, as the information will be presented quantitatively in order for it to be suitable for statistical analysis.

The rest of the article will be structured as follows. In first place, we will introduce the reader to the current background regarding the importance of a web's interface to improve user experience and how different type of variables measured nowadays can help us identify usability and functionality problems. Then we will introduce UXJs, our approach for user information tracking, stating its tracking abilities and detailing its structure. Once the approach has been introduced, we will introduce our use case scenario and show how the gathered information is presented to the web's administrator. After the detailed description and the use case, we will evaluate our system in comparison to some of the other existing tools, in order to state the similarities and differences of our system with its competence. Finally, we will state our conclusions and introduce the reader to our future work, regarding some of the functionalities to be implemented in the near future and some aspects that we consider improving.

II. BACKGROUND

The usability of a product or system, which in our current scenario would be a web page, is defined as the ''degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use'' by the ISO 25010 [32]. Also based on the information included in the standard, usability is composed of a group of sub-characteristics like learnability, operability, user error protection and accessibility, among others.

Despite of its precise definition, the measurement of a website's usability is a difficult task, as none of the existing methods seem to be definitive. As stated by Alonso-Virgós *et al.* [23], there are currently three types of usability measurement techniques: usability inspection, evaluation method and automatic evaluation. Usability inspection is done by a web usability expert and consists in the application of theoretical ideas to the web being analyzed [24], [25] whilst the evaluation method requires user participation and includes the evaluation of the user's interaction with the web page by an expert through the usage of tests, interviews and analysis of different types of parameters. The automatic evaluation approach includes tools capable of automatically evaluating the usability of the web and its contents by measuring different type of parameters that can be monitored with code sentences. Related with this last type of evaluation, the work by Santana and Baranauskas [8] states that HCI investigators consider there are two types of automatic evaluation tools: those that use the source code as the data source and those that analyze usage data in the form of logs, captured server-side or client-side. Client-side capturing is the most complex of the alternatives but enables the collection of information about the actions and the user interface elements where those actions took place; thus, client-side capturing is the most common scenario nowadays. The use of this type of tools is capable of reducing both the time and the cost of website development and maintenance, as heavy-duty tasks like log analysis can be reduced or automatized [7]. However, log analysis is a very complex task and requires a lot of processing in order for the results to be presented, so we considered investigating the suitability of different types of user interaction variables to be monitored.

Due to its importance in previous works, we considered investigating two of the key variables regarding user interaction identified by several studies: mouse and keyboard interaction. As stated by Tzafilkou and Protogeros [10], mouse and keyboard interaction monitoring 'is a vital part' of this type of scenario, as it can ' give very useful information of the perceived user experience and the system's usability'. Additionally, mouse interaction is cheaper and simpler to implement than other alternatives [9], which makes it a perfect candidate for measuring usability of web pages. Furthermore, other studies theorize about the possibility of identifying different typologies of users based on how they use the mouse [11]–[13] whilst others state that mouse interaction, regardless of the type of task and environment, can be used to model user's behavior [14]. Regarding the modelling of user's behavior, several previous studies have managed to identify and classify common mouse behavior patterns [13], [15]–[18]. Keystroke interactions are also an interesting variable, as the works by Epp *et al.* [19] have shown keyboard interactions seem to be as promising as mouse interactions for modeling user behavior. Thus, mouse interaction and keyboard interaction are two of the variables that were identified as key for measuring usability and functionality of a web page and, in such way, were included in UXJs. Although time is not directly related with usability and accessibility in previous studies, we also considered monitoring the amount of time a user spends in each of the URLs that compose a web page. Taking into account the complexity of the layout and content of a given URL, the amount of time a user spends in it can be used to identify potential usability problems.

Lately, some companies have been developing client-side user interaction collection tools to help their customers track what users do on their web pages, using the collected data to provide their clients structured information that allows them to analyze what user's do with their platforms and extract usage patterns that indicate if the corresponding web page has

some kind of design, layout or functionality problem. These tools are very similar among them in terms of structure and functionality, as all of them require the web page to include a fragment of Javascript code customized with a client API key, provided to the administrator of the web page subject of the analysis by registering an account on the tracking tool platform. Once the Javascript code is configured and linked in the web's code, the platform begins recording user interaction with the elements in the web page, as well as other kind of data like the operating system the user is working with or the amount of time the user is in session on the given web page. All of this data can be accessed by the administrator of the web page through a private area where he can see, in different types of graphical representation, the values of the different parameters being measured by the tool.

One of these tools is Google Analytics [26], developed and managed by Google, which is commonly used to track web usage information by some of the biggest web pages and applications that exist nowadays. For example, this tool can be used to measure data from e-commerce sites [3] as well as data from other type of web pages like tourism related sites [1], marketing related scenarios [2] or education products like Moodle [31]. This library, included in the web page as a Javascript component, can track different types of user interaction and presents information grouped by different parameters; with this type of structure and functionality, Google Analytics can present information in various ways, including segmentation by users and the tracking of exclusive users. Moreover, its link with other products offered by Google, such as AdWords and AdSense for example, can lead to a great amount of information regarding complex situations like the results of different online marketing campaigns. Another great functionality offered by Google Analytics is the possibility to check information on a real time basis, which allows administrators to track, on special circumstances, what is happening on their web in real time. However, there are some drawbacks to the usage of Google Analytics; for example, if the user has cookies blocked in its browser or is using private browser, the figures collected by Google Analytics won't be correct. Moreover, Google Analytics is not capable of showing specific data of the user's interaction in a given page inside the web being analyzed; this tool is only capable of tracking the amount of time the user is inside the given page but it cannot show how many interactions a button received, for example.

There are other tools like Hotjar [27], which considers itself an all-in-one tool. This consideration is based on the fact that it covers both analytics and user feedback gathering. Its main goal is to help web creators understand what visitors are trying to do with the web page. There are many interesting features in Hotjar like conversion funnels, form analysis and user surveys; however, its most important features are the following two: heatmaps and session recordings. Heatmaps are a representation of the web page where especially active parts of it are colored in order to let administrators know which parts of the web are heavily used and which ones get less interaction rates. Session recordings are video recordings of user interactions on the web, which allow administrators to directly see how people interact with the web, giving an extra support for the results shown in the heatmap. One of the main advantages of Hotjar is the clarity with which the tool represents the gathered information; in addition, it offers a large number of functionalities, which makes it one of the most complete tools on the market. Despite its great functionalities, which include some extra features not available in Google Analytics, Hotjar has also got some disadvantages. In first place, Hotjar is not mobile device friendly and in order to check all of the information it stores the use of a laptop or desktop computer is required. Its other main disadvantage is its pricing, as the tool is not free to use in professional environments; pricing depends on the number of visits or pageviews that the web being analyzed has in a day. Nevertheless, Hotjar and Google Analytics are not the only existing tools with this kind of functionality. Matomo [29] and eTracker Analytics [28] are some of the other platforms or tools existing nowadays. With similar functionalities to the ones in Hotjar and Google Analytics, Matomo and eTracker Analytics provide extensive information about user interaction in a given web page. However, the need for extra configuration on the web's code is high in order for the information to be collected correctly and measured accurately. Thus, these tools require a great maintenance effort and continuous code changes based on the objective of information gathering, scenario far from ideal for our current needs.

Another of the tools we have investigated is Open Web Analytics (OWA) [30], which is an open source web analytics software used to get information about the usage patterns inside web pages and applications. This tool, with its las stable version published in July 2018, is capable of adding web analytics functionality to websites using different programming languages like Javascript and PHP, or using REST based APIs, being also capable of integrating itself with popular content management software like WordPress. In addition to giving simple information like the total number of visits a page receives, the average visit duration and the number of unique visitors a website has, OWA is also capable of indicating other types of information like: bounce rate, browser types and operating system used by visitors, visitor geo-location and number of pages accessed per visit, among others. OWA has also got the capability of showing a top pages list, which shows the administrator the most visited pages of the site and creating a heatmap with the user interaction patterns. Just like the others alternatives previously mentioned, OWA needs some extra configuration in order to work as expected; each of the pages of the site must include several lines of code in order for the tracking to fully work and based on the type of information looking to be monitored in each of the pages, this code changes.

With this situation, our main goal was to create a tool capable of gathering and storing quantitative information about basic user interaction with all the components of a website

with no need for extra configuration on the server side. The information collected by our approach is the following: the number of visits received by each of the URLs of the web, the amount of time the user is browsing each of the URLs of the web, the amount of clicks the user is doing in each of the URLs of the web, the amount of keyboard strokes the user is doing in each of the URLs of the web, the amount of scroll (distance and travel points) the user is doing in each of the URLs of the web, the amount of clicks the user is doing in each HTML element of the URL and the amount of keyboard strokes the user is doing in each HTML element of the URL. UXJs will imitate part of the behavior of the mentioned commercial applications, as it will consist of a Javascript code included in the web and customized with an API key value and an administration site that will show all of the collected information in a comprehensive and intuitive way.

III. UXJs: OUR APPROACH

UXJs is a novel research approach for automatically collecting all possible information about the user activity in websites, showing this information quantitatively and allowing the automatic statistical analysis and the rapid understanding of the information by web developers. It is a standalone and free, Javascript oriented user interaction monitoring system designed to collect and analyze the browsing information of each of the users visiting a web page. Like the majority of the alternatives existing nowadays, UXJs is based on the creation of a unique identification token for the web page, token which will be used as a configuration parameter in one of the components of the system; this token identifies the correspondence of the interactions done by the user and the web page where these interactions took place.

UXJs consists of three separate but interconnected components, which are:

- **uxjs.js**: The Javascript library to be included inside the code of the corresponding web page. This file is responsible for the monitoring of the user's interaction inside the web page, including but not limited to the clicks done in each of the page's elements, the distance the user's mouse covers when scrolling and moving through the page and the amount of time the user spends in each of the web's sections or pages. The unique web identification token previously mentioned is located in this file.
- **UXJs API**: The API that receives the information collected by the Javascript library. This piece of the system is in charge of receiving and storing all the data collected by the Javascript library, in order for it to be presented to the administrator of the web.
- **UXJs Admin Web**: This web page presents all the gathered information to the administrator of the monitored site. When the administrator logs in, he sees a graphical summary of all the information gathered by the library and is able to filter this information per section and component; for example, the administrator can know how many users visited a given URL but can also know how many of those users clicked in a specific button.

The following diagram is a graphical representation of how the system works. Once the administrator of the web page has created the identification token and included the uxjs.js Javascript file with the functionality in the code of the page, the system is ready to record user interaction. When the user, using any type of device, requests one of the pages included in the web, the code is brought from the server and rendered by the corresponding browser, including the uxjs.js file. After the page has finished loading, the Javascript file begins to monitor user interaction, collecting all the information generated by the user's action; for example, UXJs is capable of collecting the list of points through which the mouse travels while the user is scrolling and navigating through the page or the clicks received by each of the elements in the screen. After each of these interactions, uxjs.js sends the corresponding request to the UXJs API, which stores the information received inside the system's database; however, in order to limit the amount of calls delivered to the API and have the less impact possible in the overall performance of the web page, the interactions related to mouse traveling around the page are grouped by the Javascript file and sent as a whole when the detected interaction is over. Each of the user that connects to the web page generates a new set of interactions and information, separately identified by the IP address from which these interactions are done.

At any time, the administrator of the monitored web page can access de UXJs admin web in order to check the information that is being collected and stored by the system. Once he introduces his credentials, he is presented a graphical summary of the interactions received and a results table grouped by the different URLs that exist in the site. Each of the entries of the mentioned table contains the following information: the number of visits received, the mean time the user spends inside that URL, the total time users have spent inside that URL, the mean number of actions a user does inside the given URL and the total number of actions recorded for that URL. If this information is not enough for the administrator, each of the rows of this summary table has a link to a detail view where the administrator can check, in greater detail, the interactions of each URL. The details view of each of the URLs presents the information divided into two different tables:

- **User summary:** This table collects one row for each of the users that visited the corresponding URL. The first information provided in the table is the IP address of the user, which allows to uniquely identify each of the users that visited the page. The rest of the data collected in this table includes the number of clicks the user did on the page, the number of keystrokes, the amount of mouse movements and the time the user spent in the page. There is also a place for additional information regarding the device used for browsing the page, as the table also shows the operating system of the device in question and the browser used.

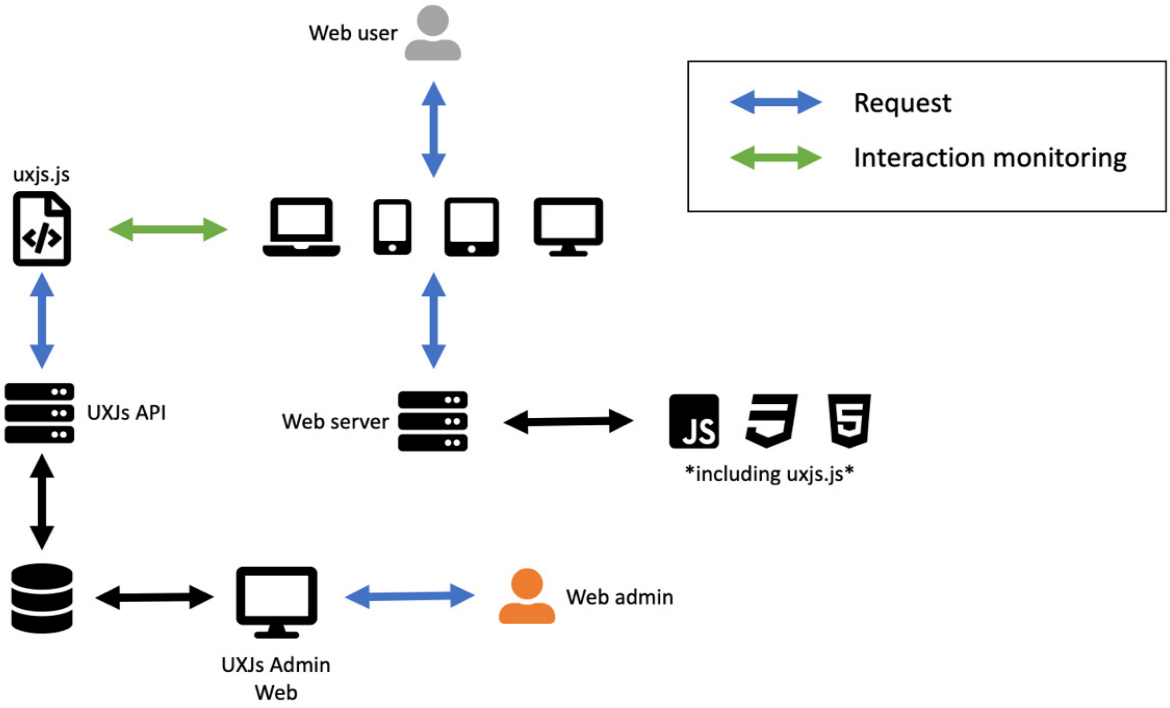

FIGURE 1. UXJs system overview.

- **Action summary:** This table collects information regarding the interactions received by different elements of the page; for example, the number of clicks received by a button or the number of input keystrokes received by a field in a form. Each of the rows of this table provides a percentage of visitors of the given URL that interacted with the corresponding element and also the mean amount of times the user interacted with the component during the session.

A. USER INTERACTION BEING MONITORED

Once the general overview of UXJs has been introduced, the next step is to explain how the interactions of the user are going to be collected throughout his browsing of the web page.

In first place we must state that in order to reduce both the number of requests to and from the API and the impact UXJs has in the performance of the web page, UXJs only sends information to the API when the current page is going to be unloaded. Each time a page is loaded, the API receives a session initialization call in order to set up the primary information related with the user: IP address, operating system, browser, etc.; once this information is set up, the UXJs library starts storing information at runtime about the different interactions the user is doing up until the time one of these interactions results in the unloading of the current page. At the exact moment when the unload event is sent, UXJs sends all the collected information for the current page to the API and when the requested page is set up, a new initialization call is made. This scenario can be seen in the following graph.

In order to provide the different results that can be seen in the administration site of the system, UXJs is monitoring the following information: the number of visits received by each of the URLs of the web, the amount of time the user is browsing each of the URLs of the web, the amount of clicks the user is doing in each of the URLs of the web, the amount of keyboard strokes the user is doing in each of the URLs of the web, the amount of scroll (distance and travel points) the user is doing in each of the URLs of the web, the amount of clicks the user is doing in each HTML element of the URL and the amount of keyboard strokes the user is doing in each HTML element of the URL.

In order to store all these types of information, we have created a generic element called action that is capable of holding the information associated to each of these types. For each measured user interaction, the system creates an action and populates each of its fields with the suitable information for the type of action to be stored. The following paragraphs will describe, in greater detail, how some of the most important parameters are being measured; specifically, we are going to describe the clicks per element, the key stroke monitoring and the mouse scroll functionalities, as these are some of the differential features of UXJs and the rest of the tools mentioned in this paper. As a finishing point for this section of the paper, we will explain how and when the storage of all the information collected is done.

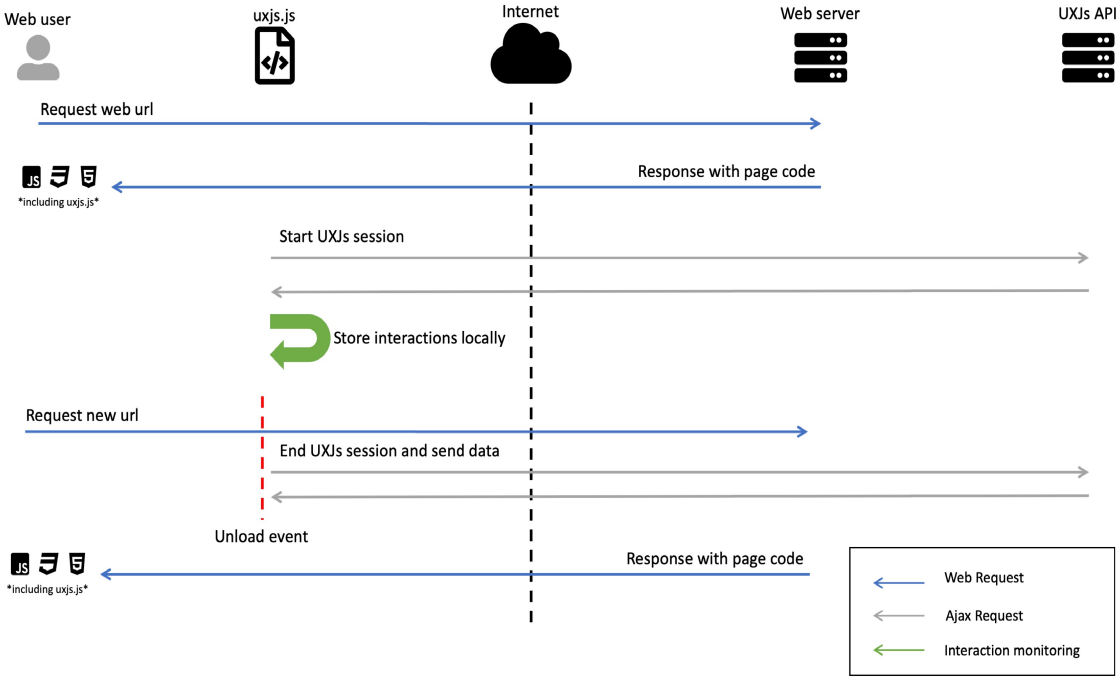

FIGURE 2. UXJs user interaction monitoring calls.

1) CLICKS PER ELEMENT

UXJs is capable of showing the number of clicks received by a specific html element inside the web; there is no restriction to the type of element the user interacted with, as all kinds of html elements (buttons, divs, spans, links, etc.) are supported. But the information regarding the clicks received by an element is not restricted to a quantitative approach, as the administration web is also capable of displaying the percentage of users who clicked that element based on the number of visits received by the page and the number of clicks received by the element in question.

In order to provide such kind of information, UXJs monitors the click events for all of the elements in the page using the document object. Once a click is detected, all of the important information regarding the event is stored as an action; this information includes the current timestamp, the position the element is in (x and y coordinates), the id attribute of the element and the html representation of both the clicked element and its parent. The recently created action is then stored in the collection that holds all of the actions the user has done in the current page, collection that will be sent to the UXJs API during the unload event as previously explained. The following code snippet illustrates the previous explanation, as it shows how the action is created and stored in the actions collection.

```
var action = {
  type: 1,
  id_element: event.target.id,
  x: event.clientX,
  y: event.clientX,
  date: new Date().getTime(),
  html: event.target.outerHTML,
  html_father:
 event.target.parentElement.outerHTML
 };
total_actions.push(action);
```

2) MOUSE INTERACTIONS: SCROLL AND TRAVEL POSITIONS

Mouse interactions, one of the other differential aspects of UXJs, is monitored using the mouse move events available for the different devices supported. In this case, the information being stored is simpler than in other kinds of interactions like clicks, as there is no need to store information about the current element or the father element. As it can be seen in the following code snippet, in this case the x and y coordinates are obtained from the event itself, as the mouse event holds the information of the movement that the user has done with the cursor.

```
var action = {
 type: 3,
 id_element: "",
 x: event.pageX,
 y: event.pageY,
 date: new DategetTime(),
 html: "",
 html_father: ""
};
total_actions.push(action);
```

With this information monitored and stored by the Javascript library, the administrator site is then capable of calculating the distance traveled by the cursor in the page as the difference between the positions associated to each of the mouse interaction events stored. Further, based both on the timestamp and the coordinates being stored with each of the interactions, the specific mouse travel can be rebuilt.

3) KEY STROKES: MEASURING USER INPUT

Measuring the user's input in the different elements of the page is one of the other differential aspects of UXJs. With this type of functionality, the administrator of the web page will be able to know if it makes sense to have a certain field inside a form or if the search engine of the web page has an acceptable usage ratio.

Key strokes are measured by listening to the key press events fired by the browsers when the user presses different letters on the keyboard. The actual keys being pressed are not stored for two different reasons: in first place, to avoid security issues regarding protected and/or personal fields like passwords and emails and on second place because they are not an important piece of information for the type of data and statistics being measured. As it can be seen in the following snippet, in this case the coordinates are also sent as empty values and no id attribute for the element is set; however, the outer html of the target element that fired the event is included in the saved information in order to identify which key strokes correspond to each of the inputs of the page.

```
var action = {
 type: 2,
 id_element: "",
 x: 0,
 y: 0,
 date: new Date().getTime(),
 html: event.target.outerHTML,
 html_father: ""
};
total_actions.push(action);
```

4) SAVING USER INTERACTION

All of the previously shown snippets, which make reference to the most important functionalities UXJs offers in comparison to the other tools and systems being analyzed, create an action based on the user interaction and then save that action in a Javascript array that keeps all the actions collected by the system.

In order to prevent a great number of requests to and from the API and also to prevent a great impact in web performance, UXJs only sends the collected information to the API when the unload event of the current page is fired. When this event is fired, UXJs sends the array of actions collected to the API through a POST request, making the API store all of the information collected regarding the current URL and the current user browsing the page. The system uses the API as the unique communication point with the database that stores the information; with this storage strategy in combination with a whitelist configuration the database can only be edited and queried through the API, ensuring there is no unwanted access to the collected data.

Once the storage of the actions is done successfully, the system then sends a session ending POST request in order to store the time when user session ends for the current URL. With this call we ensure the system is capable of tracking the amount of time each user spends in each of the existing URLs in the web. With the loading of the next URL to be visited, UXJs begins its cycle again: gets the basic information about the user (including IP address, operating system and browser), starts a new session for the current user and URL and begins storing information on the local array of actions, which will be populated until the next unload event is fired.

IV. USE CASE

After the general overview done about UXJs in the previous section, we intend to show how the tool is used in a real-life scenario. In first place, the administrator of the web must register in the UXJs admin site, indicating not only his personal information but also the main URL of the web. Once registered, he will receive the web's token and the platform will indicate him to download the Javascript file to be included in the web's code; this file must be edited in order to include the received token in the existing token variable, as shown in the snippet below.

```
//INSERT TOKEN VALUE HERE//
var uxjs\_token =
"butgrqy5u1lsh2xkjwfny";
////////////////////////////
var current_session;
var current_session_id;
var ip_address = "";
var total_actions = [];
```

Once the file has been correctly configured, it must be included in the web's code. In order to do so, the administrator must upload the uxjs.js file to the web server and add a Javascript include line at the same level were all other Javascript files are included. For example, in the current scenario this include is located in the footer of the web page. Once this is done, UXJs is ready to work and will start recording information of the users visiting the page and their interactions. From this point onwards, the administrator can login into the administrator site provided in order to check the data that has been collected.

Once the administrator logs in the site he will see a graph like the following, which shows the amount of visits the web received in a given period of time; the dates regarding this time period can be altered in order to show information from a broader or narrower time interval. As it can be seen in the image below, the graph shows a curve representing the visit counter of the web between the selected dates, showing how many users entered any of the URLs in those days.

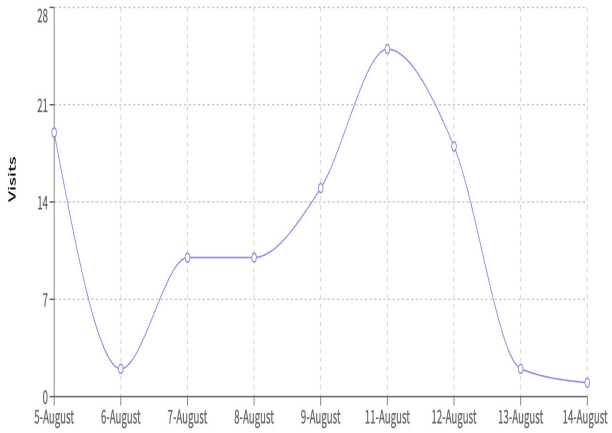

FIGURE 3. Visits graph on UXJs administrator site.

This dashboard view will also contain a simple table with some of the basic information collected by the system; this information will be segmented per URL, allowing the administrator to see which URL of his web is the most visited or which URL is the one where the users spend more time browsing. As it can be seen in the image below, the home page of this site in particular is the most visited URL of the web, followed by the active promotions and the news sections. Besides showing the number of visits, this table also shows the mean session time per user, the total session time, the mean number of actions registered by user and the total amount of actions registered. As it can be seen, despite the great number of visits received by the home page, the page where users spend most time is the news section, which seems to have a more interesting content than the other pages.

With this kind of information resumed in a table like the one presented below, the administrator of the website is capable of extracting some quick information about his site. For example, the administrator can clearly see which is the most visited page of the site or which is the page with the most actions registered; however, one of the most valuable pieces of information the administrator of the site is capable of extracting from this kind of table is the pages where potential problems could surface. Thus, if a given page has a very high mean session time or a high value for the mean number of actions this could mean there is some kind of usability problem with the page and the users are finding difficulties to use it. There could also be some kind of usability problems with pages where the users spend little time or interact little with its elements. In the figures represented in the table below, it can be seen that the contact section of the monitored web has some kind of difficulties as there are very little visits on the page and no interactions recorded, which could mean the users are having difficulties reaching or using the existing contact form.

If the administrator needs to know detailed information for one of the existing URLs, the details button at the end of each row of the previously shown table gives him access to the

IP Address	Type 1 actions	Type 2 actions	Type 3 actions	Browser	Operating system
85.152.6.13	1	0	122	Chrome 75	Windows
46.24.243.10	2	0	387	Chrome 75	Windows
18.191.89.137	0	0	0	Chrome 75	Linux
31.4.197.179	1	0	1	Chrome 67	Android
2.136.63.248	1	0	1	Chrome 75	Windows
79.109.91.86	1	0	1	Chrome 76	Android
85.152.151.87	0	0	0	Firefox 68	Windows
176.86.112.217	0	0	0	Safari 12	Mac OS
31.4.138.43	1	0	1	Chrome 73	Android
85.152.140.185	7	0	8	Chrome 59	Android
IP Address	Type 1 actions	Type 2 actions	Type 3 actions	Browser	Operating system

FIGURE 4. URL basic information details.

dashboard view of each of the pages. One of the tables present in this dashboard view is shown below; for this example, the table below shows the information about user sessions in the home page of the web site. As it can be seen, this table has a different layout, as the information shown is per user: IP address, number of interactions of each type, browser and operating system. For example, in this table we can see that there are no keystroke actions stored (type 2 actions) but that nearly all of the users click, at least once, on an element of the page (type 1 actions). Information about the different browsers and operating systems is also included, giving the administrator some extra information that in the future could be used to segment the results by platform and device, as it will be stated in our future work.

URL	Total visits	Mean session time (HH:mm:ss)	Total session time (HH:mm:ss)	Mean actions registered	Total actions registered	...
/	37	00:00:17	00:10:38	28	1023	Details
/active-promotions/	21	00:01:41	00:35:29	62	1299	Details
/news/	11	00:02:06	00:23:07	21	226	Details
/about-us/	8	00:00:47	00:06:18	12	94	Details
/news/perez-sala-11-sold-out/	7	00:00:12	00:01:23	1	8	Details
/finished-promotions/	6	00:01:27	00:08:42	45	269	Details
/news/current-state-ps11-works/	5	00:00:55	00:04:33	11	53	Details
/contact/	2	00:00:01	00:00:03	0	0	Details
/news/current-state-perez-sala-11/	1	00:00:15	00:00:15	0	0	Details
/news/amancio-ortega-pontegadea/	1	00:00:54	00:00:54	2	2	Details
URL	Total visits	Mean visit time (HH:mm:ss)	Total visit time (HH:mm:ss)	Mean actions registered	Total actions registered	...

FIGURE 5. URL details view.

V. EVALUATION

Throughout the paper, we have been introducing all the functionalities and possibilities UXJs offers and how these features enable us to measure users' behavior and interaction with the different pages and components of our website. With the layout and content of the following table we intend to clearly establish all the use cases our system is capable of tracking and how other alternatives manage them. _

The first column of the table indicates the type of user activity monitored, from the amount of time each user browses each of the pages in the website to the number of clicks and keyboard strokes each of the html elements of the web (inputs, buttons, divs, etc.) receives. On each of the remaining columns we establish if each of the mentioned tools is capable of tracking and recording the given activity; there are three possible values for each cell: yes, no or yes*,

User Activity	Google Analytics	Hotjar	eTracker Analytics	Matomo	Open Web Analytics	UXJs
Browsing time in each page	Yes	No	No	Yes*	No	Yes
Visit counter in each page	Yes	No	Yes	Yes*	Yes*	Yes
Clicks in each page	No	No	No	No	Yes*	Yes
Keyboard strokes per page	No	No	No	No	No	Yes
Mouse scroll and travel per page	No	No	No	No	No	Yes
Clicks in each html element of the page	No	No	Yes*	Yes*	Yes*	Yes
Keyboard strokes in each html element of the page	No	No	No	No	No	Yes

with this last value representing the need for additional configuration on the web's code.

As the table illustrates, none of the existent competence for UXJs can deliver quantitative information about all interactions done by the user inside a web page without the need for extra configuration in one or both of the sides. For example, Hotjar can provide a heatmap with the most interacted zones inside a given page and can also provide recordings of what the users do inside that page but is not capable of providing a quantitative summary of those interactions. On the other hand, Google Analytics is capable of giving information about the number of visits a given page has and also how much time a user browses through the page, but cannot provide information of its interactions inside it. For example, in order to know the amount of users that used a given form inside the web page with Google Analytics, the administrator would need to monitor the amount of times the thank you page has been rendered with a given parameter that indicates which was the form used; on the other hand, UXJs is capable of providing this information by recording the amount of clicks the submit button of the mentioned form has been clicked by users, without the need for any type of parameters or additional configurations on the thank you page.

As the table shows, some of the functionalities do exist on tools like Matomo, Open Web Analytics and eTracker Analytics but require additional configuration on one or all of the code's sides. For example, Matomo can track the clicks in each of the html elements inside a web page but the code of the web page must be prepared to do so: in order for this functionality to work, each of the elements to be tracked must contain specific classes and/or attributes in order for the Matomo platform to record the user's interaction. Open Web Analytics is a similar case to Matomo's, as an additional piece of JavaScript code must be included in each of the pages' code in order for the system to track the desired actions: the tracker must be created and initialized per page basis and within its initialization the type of interactions to be monitored must also be stated. Thus, in each of the pages where a specific interaction should be monitored, there is the need to modify the code and include the desired configuration for the tracker.

On the other hand, UXJs' interaction tracking is done transparently. For example, the tool is capable of identifying the clicked element by its id and by the code of its father element, avoiding the need for special classes, custom attributes and/or specific Javascript sentences that allow the identification of the element and enable click tracking on it. In order for UXJs to track the information on a given page, the only need is the importation sentence for its Javascript file; once the file is imported, the tool is capable of tracking all the information regarding the user's interactions without the need for any changes on the web's source code. This reduces the complexity of integrating the tracking system to the minimum possible.

As it can also be seen in the table, some of the functionalities do exist on tools like Matomo and eTracker Analytics but require additional configuration on one or all of the code's sides. For example, Matomo can track the clicks in each of the html elements inside a web page but the code of the web page must be prepared to do so; in order for this functionality to work, each of the elements to be tracked must contain specific classes and/or attributes in order for the Matomo platform to record the user's interaction. With UXJs, click tracking is done transparently as the tool is capable of identifying the clicked element by its id and by the code of its father element, avoiding the need for special classes and/or custom attributes that allow the identification of the element and enable click tracking on it.

VI. CONCLUSION AND FUTURE WORK

Usability and accessibility are two of the key aspects related to a web page nowadays and a good balance between them is mandatory in order for the page to offer a good service to its users. Measuring web usability is a complex task and none

of the existing methods seems to offer a definitive measuring method, as they measure information based on specific usage scenarios or domains, and do not always present information in a usable way. Existing approaches like Google Analytics, Hotjar or Matomo collect user interaction information like the number of visits and conversion ratios but other important information (for example, mouse and keyboard interaction) is not stored.

Thus, in this paper we present a novel research approach for automatically collecting all possible information about the user activity in websites called UXJs. UXJs integrates seamlessly with the code of the website and monitors the interactions (clicks, keyboard interactions and mouse interactions) the website user does during its visits to each of the pages of the website, without the need for any kind of configuration or tweak of the website's code. This approach is capable of showing the collected information quantitatively and allows the automatic statistical analysis and the rapid understanding of this data by web developers.

Nevertheless, UXJs is in an initial state and further work must be done in order for it to be considered a finished tool. The first point of our future work is centered on giving additional data analysis to the administrators in the admin site; for example, the creation of a report view with detailed data analysis and upgrade recommendations based on the information gathered in the different pages of the web. Other of our future work plans is to segment information based on the type of device used to access the web, in order to allow a comparison between the interactions a given component receives when the page is rendered in a desktop scenario versus a mobile scenario, pointing out potential problems related with mobile design and layout. We are also looking into the creation of a comparison index based on the different types of information gathered by the tool. This comparison index will enable the creation of a direct comparison between two different layouts for the same information and stablish a type of A-B testing scenario where the results are directly obtained from user interaction figures.

REFERENCES

[1] B. Plaza, "Google analytics for measuring website performance," *Tourism Manage.*, vol. 32, no. 3, pp. 477–481, Jun. 2011.
[2] J. Järvinen and H. Karjaluoto, "The use of Web analytics for digital marketing performance measurement," *Ind. Marketing Manage.*, vol. 50, pp. 117–127, Oct. 2015.
[3] L. Hasan, A. Morris, and S. Probets, "Using Google analytics to evaluate the usability of E-commerce sites," in *Proc. 1st Int. Conf. Hum. Centered Design, Held HCI Int.* San Diego, CA, USA: Springer-Verlag, 2009, pp. 697–706.
[4] G. Brajnik, "Using automatic tools in accessibility and usability assurance processes," in *Proc. 8th ERCIM UIALL Workshop*, in Lecture Notes in Computer Science. Springer-Verlag, 2004, pp. 219–234.
[5] J. Abascal and C. Nicolle, "Moving towards inclusive design guidelines for socially and ethically aware HCI," *Interacting Comput.*, vol. 17, no. 5, pp. 484–505, Sep. 2005.
[6] F. Correani, B. Leporini, and F. Paternoò, "Automatic inspection-based support for obtaining usable Web sites for vision-impaired users," *Universal Access Inf. Soc.*, vol. 5, pp. 82–95, May 2006.
[7] L. Paganelli and F. Paternò, "Intelligent analysis of user interactions with Web applications," in *Proc. 7th Int. Conf. Intell. User Interfaces (IUI)*, New York, NY, USA, 2002, pp. 111–118.
[8] V. F. D. Santana and M. C. C. Baranauskas, "WELFIT: A remote evaluation tool for identifying Web usage patterns through client-side logging," *Int. J. Hum.-Comput. Stud.*, vol. 76, pp. 40–49, Apr. 2015.
[9] E. Arroyo, T. Selker, and W. Wei, "Usability tool for analysis of Web designs using mouse tracks," in *Proc. ACM CHI Conf. Hum. Factors Comput. Syst.*, vol. 2, 2006, pp. 484–489.
[10] T. Katerina and P. Nicolaos, "Mouse behavioral patterns and keystroke dynamics in end-user development: What can they tell us about users' behavioral attributes?" *Comput. Hum. Behav.*, vol. 83, pp. 288–305, Jun. 2018.
[11] A. L. Leiva and V. R. Hernando, "A gesture inference methodology for user evaluation based on mouse activity tracking," in *Proc. IADIS Int. Conf. Interfaces (HCI)*, 2008, pp. 18–26.
[12] K. Rodden and X. Fu, "Exploring how mouse movements relate to eye movements on Web search results pages," in *Proc. WISI Workshop*, 2007, pp. 1–64.
[13] K. Rodden, X. Fu, A. Aula, and I. Spiro, "Eye-mouse coordination patterns on Web search results pages," in *Proc. 26th Annu. CHI Conf. Extended Abstr. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2008, pp. 2997–3002.
[14] Z. Hinbarji, R. Albatal, and C. Gurrin, "Dynamic user authentication based on mouse movements curves," in *Proc. 21st Int. Conf. Multimedia Modeling (MMM)*, Sydney, NSW, Australia: Springer, Jan. 2015, pp. 111–122.
[15] R. Atterer, M. Wnuk, and A. Schmidt, "Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction," in *Proc. 15th Int. Conf. World Wide Web (WWW)*, Edinburgh, U.K., 2006, pp. 203–212.
[16] S. Ferreira, E. Arroyo, R. Tarrago, and J. Blat, "Applying mouse tracking to investigate patterns of mouse movements in Web forms," Univ. Pompeu Fabra, Barcelona, Spain, Tech. Rep., 2010.
[17] G. Lee and Z. Chen, "Investigating the differences in Web browsing behaviour of Chinese and European users using mouse tracking," in *Usability and Internationalization. HCI and Culture* (Lecture Notes in Computer Science), vol. 4559, N. Aykin, Ed. Berlin, Germany: Springer-Verlag, 2007, pp. 502–512.
[18] F. Mueller and A. Lockerd, "Cheese: Tracking mouse movement activity on websites, a tool for user modeling," in *Extended Abstracts on Human Factors in Computing Systems*. Seattle, WA, USA: ACM Press, 2001.
[19] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *Proc. Annu. Conf. Hum. Factors Comput. Syst. (CHI)*, Vancouver, BC, Canada, May 2011, pp. 715–724.
[20] C. A. Gumussoy, "Usability guideline for banking software design," *Comput. Hum. Behav.*, vol. 62, pp. 277–285, Sep. 2016.
[21] A. Sivaji, N. Abdollah, S. S. Tzuaan, C. N. Khean, Z. M. Nor, S. H. Rasidi, and Y. S. Wai, "Measuring public value UX-based on ISO/IEC 25010 quality attributes: Case study on e-Government website," in *Proc. 3rd Int. Conf. User Sci. Eng. (i-USEr)*, Sep. 2014, pp. 56–61.
[22] J. Nielsen. *10 Usability Heuristics for User Interface Design*. Accessed: Mar. 3, 2020. [Online]. Available: http://www.nngroup.com/articles/ten-usability-heuristics/
[23] L. Alonso-Virgós, J. P. Espada, and R. G. Crespo, "Analysing compliance and application of usability guidelines on efficient and understandable controls," *Comput. Standards Interfaces.*, vol. 66, Oct. 2019, Art. no. 103349.
[24] C. Mariage, J. Vanderdonckt, and C. Pribeanu, "State of the art of Web usability guidelines," in *The Handbook of Human Factors in Web Design*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 2005, ch. 41, pp. 688–700.
[25] J. Nielsen and H. Loranger, *Prioritizing Web Usability*. Thousand Oaks, CA, USA: New Riders, 2006.
[26] *Google Analytics*. Accessed: Mar. 3, 2020. [Online]. Available: https://analytics.google.com/analytics/web/
[27] *Hotjar*. Accessed: Mar. 3, 2020. [Online]. Available: https://www.hotjar.com/
[28] *ETracker Analytics*. Accessed: Mar. 3, 2020. [Online]. Available: https://www.etracker.com/en/analytics/
[29] *Matomo*. Accessed: Mar. 3, 2020. [Online]. Available: https://matomo.org/
[30] *Open Web Analytics*. Accessed: Mar. 3, 2020. [Online]. Available: http://www.openwebanalytics.com/
[31] S. J. Turner, "Website statistics 2.0: Using Google analytics to measure library website effectiveness," *Tech. Services Quart.*, vol. 27, pp. 261–278, May 2010, doi: 10.1080/07317131003765910.
[32] *Software Quality Model*, Standard ISO/IEC 25010, International Organization for Standardization, Geneva, Switzerland, 2011.

JAIME SOLÍS-MARTÍNEZ received the Ph.D. degree in computer engineering from the University of Oviedo. He is currently an Associate Professor with the Computer Science Department, University of Oviedo. His research interests include model driven engineering, domain specific languages, business process modeling and the application of these technologies to web and mobile device applications, and exploration of human–computer interaction and usability issues.

JORDAN PASCUAL ESPADA received the Ph.D. degree in computer engineering from the University of Oviedo. He is currently a Lecturer with the Computer Science Department, University of Oviedo. His research interests include the Internet of Things, exploration of new applications and associated human–computer interaction issues in ubiquitous computing and emerging technologies, particularly mobile and Web.

RUBÉN GONZÁLEZ CRESPO received the Ph.D. degree in software engineering, industrial engineering, and computer engineering from the Pontifical University of Salamanca. His research interests include mobile devices, multiagent systems, MED, accessibility, and software engineering. He was awarded with honors in the Doctoral Research.

B. CRISTINA PELAYO G-BUSTELO received the Ph.D. degree in computer engineering from the University of Oviedo. She is currently a Lecturer with the Computer Science Department, University of Oviedo. Her research interests include object-oriented technology, Web Engineering, eGovernment, and modeling software with BPM, DSL, and MDA.

JUAN MANUEL CUEVA LOVELLE received the Ph.D. degree from Madrid Polytechnic University, Spain, in 1990.

He was a Mining Engineer with the Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). Since 1985, he has been a Professor with the Languages and Computers Systems Area, Oviedo University, Spain. His research interests include object-oriented technology, language processors, human–computer interface, Web engineering, and modeling software with BPM, DSL, and MDA. He is an ACM and IEEE Voting Member.