Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction

Richard Atterer
Media Informatics Group
University of Munich
Amalienstr. 17
80333 Munich, Germany
richard.atterer@ifi.lmu.de

Monika Wnuk
Media Informatics Group
University of Munich
Amalienstr. 17
80333 Munich, Germany
wnukm@ifi.lmu.de

Albrecht Schmidt
Embedded Interaction
Research Group
University of Munich
Amalienstr. 17
80333 Munich, Germany
albrecht.schmidt@acm.org

ABSTRACT

In this paper, we investigate how detailed tracking of user interaction can be monitored using standard web technologies. Our motivation is to enable implicit interaction and to ease usability evaluation of web applications outside the lab. To obtain meaningful statements on how users interact with a web application, the collected information needs to be more detailed and fine-grained than that provided by classical log files. We focus on tasks such as classifying the user with regard to computer usage proficiency or making a detailed assessment of how long it took users to fill in fields of a form. Additionally, it is important in the context of our work that usage tracking should not alter the user's experience and that it should work with existing server and browser setups. We present an implementation for detailed tracking of user actions on web pages. An HTTP proxy modifies HTML pages by adding JavaScript code before delivering them to the client. This JavaScript tracking code collects data about mouse movements, keyboard input and more. We demonstrate the usefulness of our approach in a case study.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Evaluation/methodology*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Interaction styles*; D.2.5 [**Software Engineering**]: Testing and Debugging; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*User issues*

General Terms

Experimentation, Human Factors

Keywords

Website usability evaluation, implicit interaction, mouse tracking, user activity tracking, HTTP proxy

1. INTRODUCTION

In recent years, the web has constantly been gaining importance as a platform for applications. Whereas earlier web applications were simple and used straightforward page layouts, now websites offer applications with sophisticated user interfaces. Additionally, there is a noticeable tendency to move more of the application to the client: Earlier web applications followed the HTTP protocol's request/response paradigm, but many newer applications are JavaScript-based and only contact the server in order to load or save data.

These developments pose a problem when it comes to obtaining feedback about the usage of web applications. The data left by many interactive applications in the server's log file is minimal and not sufficient for extracting detailed information about the actual usage of the application. For instance, it is not possible to say in which order the fields of a form were filled in.

In this work, we investigate means for obtaining more information about the usage of websites and web applications. This includes a detailed tracking of all interaction with the displayed browser page, such as moving the mouse pointer around or scrolling the page. Additionally, the interaction should be tracked at the widget level, i.e. the mouse coordinates are mapped to elements like buttons, links etc. Combined with knowledge about the layout of the HTML pages, complete tracking of all user activity becomes possible.

The information which can be obtained using activity tracking is interesting for a number of scenarios. So far, the main application has been usability tests of websites, but with a tracking approach that is flexible enough, it is also possible to use the tracking during web application development, beta-testing or constant/repeated evaluation of live websites. On a more abstract level, it can be employed for profiling users and for implicit interaction with websites.

The term "implicit interaction" is explained in detail in section 3.2. In contrast to explicit interaction with a website which the user is aware of (such as filling in a form field), implicit interaction usually happens unconsciously. For example, the user may hesitate before filling in the field because he is not certain about the correct answer – a fact which we can observe and make use of.

As part of our research, we present an advanced, non-intrusive tracking solution which does not require special setup at the client or server side. Working as an HTTP proxy, the software performs detailed user activity tracking while modifying HTML pages "on the fly" before passing them on to the client browser.

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

This paper is organised as follows: In section 2, our requirements for user tracking and the chosen approach are described. Next, section 3 discusses the types of data that can be obtained using activity tracking as well as its use for implicit interaction and usability testing. In section 4, the implementation of our HTTP proxy for activity tracking is explained, followed by a case study in section 5 which illustrates its usefulness. Section 6 discusses previous work, it is followed by the conclusion in section 7.

2. OVERVIEW: AN APPROACH TO TRACKING USER ACTIONS ON WEB PAGES

In this section, we will outline our requirements for a flexible, non-invasive tracking technology for web page usage, and briefly describe the approach taken by our solution. Our overall goal was to design a tracking solution which records detailed data for the analysis of user actions, without many of the restrictions of existing tools. Section 4 discusses the implementation in detail.

General requirements The general requirements for the user tracking approach are as follows. The technical requirements resulting from these are listed in section 4.1.

- Detailed tracking of user actions, such as navigation between pages, actions on a page (mouse movements, scrolling), and any input provided to the browser (clicks, text input).
- Platform independence both from the server technology and the client operating system/browser
- Transparent operation – the user's browsing experience should not be altered in any way.
- As few client-side changes as possible: In order to be useful in a wide variety of cases, user tracking should be possible without special equipment or preparation at the client side.
- As few server-side changes as possible: The server for which user actions need to be tracked might belong to someone other than the person doing the tracking, so one should not assume that changes to the server-side setup are possible. Furthermore, standard web server logs do not usually provide sufficient data for detailed tracking.
- More automation: For web usability tests, this will reduce the costs of the test (see section 3.1.1). Other usage scenarios of our approach (e.g. self-adapting websites) are not possible if manual steps are necessary.

Approach: A proxy for user tracking Some of the requirements above may seem contradictory; the tracking *must* take place either at the client or server end, which implies significant changes in the setup. However, our proposed solution addresses this issue simply by allowing the tracking to take place *either* client-side or server-side – this way, the type of setup can be chosen on a case-to-case basis depending on the task for which the tracking is employed.

In order to collect data about users' actions, we use the approach of an HTTP proxy which is inserted between the client and server (see figure 1). It intercepts all traffic and outputs log data with details about any requests sent to servers as well as the replies that a server sends back to the client.

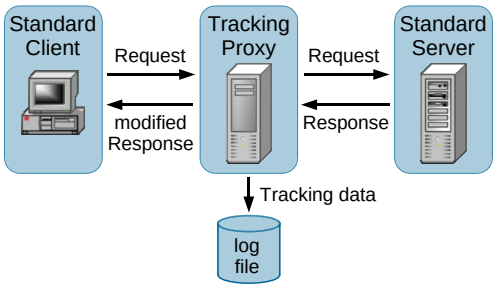

Figure 1: The HTTP proxy for activity tracking augments all HTML with JavaScript to monitor mouse movements and other types of interaction.

However, if only the requests were logged, the data collected by the proxy would be comparable to standard web server logs. Only things like the URLs of requested pages and input data typed into forms by the user would be available. This is not sufficient for our purpose of logging the user actions in detail, e.g. including mouse movement.

Thus, the next step towards a complete solution is the addition of client-side code which records the user's behaviour in detail. The most straightforward way to achieve this would be via special software running on the client machine. However, installation of software is not an option because we also want to support setups where the user does not need to change his configuration at all.

Luckily, current browsers offer flexible ways to run client-side code, in the form of the built-in JavaScript support. If we are able to run JavaScript inside the browser, detailed data about any mouse and keyboard input is available, including position of the mouse pointer, keypresses, browser window size and much more.

In a human computer interaction class at University of Munich in spring 2005, we had students manually implement detailed tracking for pages that contained larger forms. This assignment showed that it is feasible to implement this kind of tracking without altering the user's experience.

The browser's security model prevents that any JavaScript code has access to all pages displayed by the browser; only the JavaScript code of the currently displayed page is executed. For this reason, we use the following approach to have JavaScript code of our choice executed for each and every page that the browser requests: Before our HTTP proxy passes on HTML data from the server to the client, the HTML is modified. The modification causes the JavaScript part of our implementation to be loaded in the context of the page. As long as the page is displayed, the JavaScript code collects data and sends it to the proxy for logging.

In practice, a number of problems needs to be addressed to get the tracking to work in a satisfactory way. See section 4.2 for a discussion of the implementation, including issues such as how to transmit the log data back to the proxy, or how to avoid that the proxy's JavaScript interferes with other JavaScript used on the page.

3. MAKING USE OF THE COLLECTED USAGE INFORMATION

When information about user actions is recorded, it is always very concrete (e.g. "the left mouse button was clicked on link x"). This means that more abstract information must be inferred from the concrete information. In general, the following types of data can be collected with our user tracking solution:

- "Small and concrete": Information which is usually collected directly by tools, or trivial to extract from the data collected. For example, this includes information on what input field was filled in, the time required by a user to type in his own name, etc.

- "Small and abstract": Information about a user action which can only be deduced indirectly from a log. For instance: "The user probably did not read the form properly, because it was completed far too quickly", or: "The user seems to have had trouble deciding on an action, as he repeatedly hovered the mouse over the alternatives."

- "Large and concrete": Information about the user which is not just valid for a single user test in which he participates. For example: "The user is very precise when clicking on targets, but very slow when typing."

- "Large and abstract": General information about the user, such as the level of his computer skills.

Our approach of using a HTTP proxy is flexible enough to be used in a variety of different scenarios. The following is only a selection of possible areas of application:

- User profiling

- Development or debugging of web applications

- Usage analysis of websites, e.g. determining detailed usage patterns for purposes of marketing, business process streamlining or similar

- Usability tests of websites – some imaginable scenarios are described in section 3.1

- Self-adapting websites, i.e. websites which adapt some of their content (menu structure, main text on front page) to the varying demands of users

Many other applications can be envisioned. For example, an advanced form of customer support for websites is possible: If a customer has a problem using a web application, they can contact a support hotline, either by telephone or over the Internet, using an instant messaging functionality built into the application's website. Customer support is able to perform live monitoring of all actions of the user to understand the problem – even a live replay of the customer's mouse movements would be feasible.

In the following sections, we will look at two areas of application in greater detail: Usability evaluation of websites and implicit interaction with websites.

3.1 Usability Evaluation of Websites

So far, usability evaluation for websites has been the main purpose of systems which are similar to our approach. Section 6 gives an overview of related work, and compares the different available tools. In this section, we show that our solution is well suited for this task. In fact, it is flexible enough to support many different usability testing scenarios, whereas other tools usually concentrate on only one scenario, such as testing a specific, prepared website in a usability lab on an appropriately configured computer.

3.1.1 Motivation: Lowering the Cost of Testing

An expert who conducts a usability test is typically confronted with the following dilemma: On one hand, as much data as possible needs to be logged in order to produce reliable statistics. This implies inviting a large number of test users and using advanced technology (e.g. video recordings or eye tracking) during the test. On the other hand, the amount of money, time and manpower available for tests is limited. This is especially true for commercial suppliers of usability testing expertise.

Due to this dilemma, the goal when designing our usability proxy was to reduce the cost of activities which are not directly related to actually performing the test. With this in mind, we had a look at the typical tasks that normally arise when performing a 'classical' usability test:

- Before the test, a computer must be set up as a test machine. For detailed tracking, additional software and equipment like cameras, eye tracking etc. must be installed.
  Furthermore, the users must come to the test setup and meet the expert performing the test. For websites with an international audience, this makes usability testing very expensive: Either the test users or the expert and his equipment needs to travel to locations on different continents.

- During the test, usually only one user can perform the test at a time, either due to lack of equipment or because it is difficult for the expert to supervise more than one user simultaneously.

- After the test, the data obtained from cameras, eye tracking, mouse tracking, manual notes etc. must be aggregated, interpreted and summarised. In practice, our observation is that due to the sheer amount of data that was obtained, some of it is often ignored during this stage. For example, video recordings of the test are often only consulted for a few special cases where a test user's behaviour was strange or very different from the usual behaviour.

In order to reduce the cost of website usability evaluation, we decided that our user action tracking approach should have the following properties:

No test lab necessary In the discussion above, it quickly becomes clear that much of the cost is caused by the fact that the test user, usability expert and the equipment need to come to the same place for the test. This is not always necessary, because the Internet can be used for remote usability tests. In those cases where remote tests

are not sufficient, it is often possible only to invite a certain portion of users to the lab.

No special hardware or software requirements While additional video footage or eye tracking data helps to identify some problems more easily, its usefulness must always be weighed against its costs. We believe it is acceptable to do without special hardware support in many cases – see section 6.2 for a more detailed discussion. Furthermore, user tests can be parallelised much better (i.e. performed by several test users at the same time) if it is not necessary to use special equipment.

If no special software needs to be installed, the amount of technical problems can be reduced: The expert may not be present in person to install special software, whereas the test user cannot be expected to have sufficient computer skills.

3.1.2 Scenarios to Use Our Tracking Approach for Usability Evaluation

Our HTTP proxy can be used in different ways to perform usability tests:

"Classical" usability evaluation Some of the existing tools are only designed for this type of user test: A test user is told to perform certain tasks on a set of web pages. Often, the content of the pages is static, and the usability expert has full control over the server from which they originate.

Evaluation of interactive sites Due to the proxy approach, it is possible to evaluate websites which are not under the control of the usability expert, and to have test users interact with other, unknown users on these sites. For example, this includes the monitoring of online auctions or real purchases in online shops. (However, it should be noted that our implementation does not support encrypted HTTP connections at the moment, which will prevent its being used with some of these sites.)

Evaluation of highly dynamic web pages Recently, the number of sophisticated JavaScript-powered web applications has increased constantly. This type of application, where much interaction happens at the client side only, has also been referred to under the acronym AJAX (Asynchronous JavaScript and XML). Since our proxy does not interfere with any JavaScript already in use on web pages, tracking user interaction works very well. This is especially important because server log file analysis will fail completely with this scenario.

Parallel user tests The performance of our solution is sufficient to allow its use by many test users in parallel. This allows the collection of much more data than with setups where only one user can take part in the test at a time.

Remote user tests Related to the previous point, the test users can take part in the experiment from their usual desktop machine at home or work. For websites with an international audience, the test users can be located all over the world.

Inviting test users over the Internet Instead of giving the website URL to a number of users and telling them to perform tasks, it may sometimes be desirable to have a website's real users participate in a test. For this scenario, our proxy needs to be run on the website server. With appropriate server configuration (e.g. using the Apache server's mod_proxy module), the proxy can be used to track a user's actions on the website once this user has agreed to the user test, e.g. by clicking on a button. This way, the user does not even need to reconfigure his browser. Only some users (rather than all site visitors) can be tracked, and no change is necessary to the HTML code on the server.

3.2 Implicit Interaction

How people interact with an application provides additional information. Typing speed or pointing precision may be good indicators on the proficiency level of a user. The time a user spends on a specific question in a questionnaire may indicate that this question makes her think a lot. Such information is not provided on purpose by the user. However, operating an application, clicking on a button or filling in a form will inevitably provide such parameters. Up to now, they are largely ignored in classical computer systems with graphical user interfaces.

In the area of context-awareness [7] [9] and physical user interfaces, the notion of implicit human computer interaction is well established [8]. It is defined as the behaviour and interaction of a user with the environment and artifacts to reach a goal. In this work, the focus is on interaction beyond the computer with the real world. The notion of implicit interaction can be extended back to the traditional computer system. People often use applications as tools to achieve a goal, without focussing on the interaction with the computer. For example, if someone orders cinema tickets on a web page (and provided the page is well engineered), she will not think consciously where to click or which form field to fill in first. These minimal interactions will happen unconsciously and automatically.

In the context of web applications, we extend the notion of implicit interaction to the following: Implicit interaction is the observable interaction behaviour of the user with an application that is not done consciously while focusing on reaching a goal.

It is obvious that no clear distinction between implicit and explicit interaction is possible. Figuring out what is done consciously is not straightforward. However, our experience shows that users do much of the interaction of a website or in a form without thinking much about the small steps in the interaction process, and hence having the notion of implicit interaction helps to assess usage. This information can be used in the small (e.g. finding additional information needs with regard to a form field) as well as in the large (estimating the type of user).

In the following example, the concept of implicit interaction is illustrated: A user wants to buy a flight ticket at an online travel agency. Typically, the user fills in a form with the origin and destination and provides the dates of travel. That is essentially all the information that is sent back to the server. For booking a flight, this is sufficient. However, additional information on how the user filled in the form is lost. This information is not essential to book the flight, but could be used to improve the service or to customise it.

If implicit interaction tracking is present, it can detect that the user changed dates more than once before submitting the form. This may indicate that he or she is not yet sure about the date, so it can be an option for the website not only to provide flight information for the selected date, but also for earlier and later days.

Implicit input from the user does not provide clear information, it acts more as an additional source of information. The reason why a user types in a certain field much slower than the others may be due to the fact that he or she needs to think longer on this question, but it could also mean that the user needed to answer the phone while doing the question. The developer has to be aware that the additional information needs to be used carefully. Similarly, implicit interaction can be used to monitor usability over a longer period of time. This allows us to continuously look at where people stop using an application (e.g. which form field makes them go away from a website) or to identify part of an application where users are slowed down and where help may be needed.

Collecting and using implicit input raises privacy concerns, ethical questions, and to some extent legal issues. When analysing the information about how precise a user clicks, how often she needs to correct an input field or how long it takes her to write her name, do we have to treat this data as personal information? We think that this is the case and therefore we designed our system in such a way that users usually have to explicitly opt in to use the proxy server by reconfiguring their browser. In a setup where the proxy runs at the server's end, we strongly recommend that a user be asked for his approval before tracking his actions. The user should also be informed in detail about the type of data that will be collected. In our opinion, not doing do so would not only be unethical, but even illegitimate in some jurisdictions.

4. USAPROXY: AN HTTP PROXY FOR WEBSITE USAGE TRACKING

In this section, we describe the UsaProxy application that provides website usage tracking functionality using a HTTP proxy approach. First, we will look at the general requirements from section 2 at a more technical level. Next, we show how the implementation of UsaProxy meets these requirements. This is followed by a discussion of methods for the visualisation of the aggregated tracking data.

4.1 Technical Requirements

Our primary objective was to enable automated tracking of user activity on web pages that is not perceptible by the user nor intrusive. This requires an application which operates transparently in a way which does not alter the user's browsing habits and experience. It must be possible for test users to take part in website evaluation remotely from their home/office environments, from any location and using their own equipment and network access. For this purpose, the core proxy application as well as the tracking client-side part must be platform-independent with regard to the server technology, and compatible with major browsers and operating systems at the client side. Furthermore, the system must not require any installation on the client machine

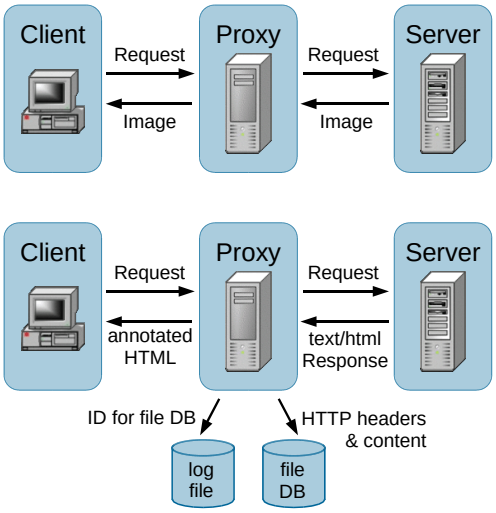

Figure 2: *Top:* Images and other data are passed on unmodified. *Bottom:* For HTML content, special JavaScript is added for the client. The server response is recorded and identified via a logged ID.

or changes to the technology used to create the pages. All these requirements are met by the HTTP proxy approach as described in section 2 and below.

For the purposes of a usability engineer, the monitoring and logging of user actions must proceed accurately, in real time and in a reproducible way. It is important to provide a range of features and possibilities that is comparable to those of a standard usability test, resulting in an accurate listing of what the user actually did while performing the predefined tasks. In order to be able to reconstruct website usage in a qualitatively adequate way, we decided to track the following user actions:

- Navigation behaviour metrics across multiple websites, such as the most frequently used path through a website, key entry and exit pages, and page views
- Time-based metrics such as the average time spent on a page, amount of time the mouse pointer hovers over elements (to detect whether the user hesitated over other interesting links/text areas before he clicked), and time of usage of elements (time taken for form fields, click frequency, general typing speed)
- Actions of a user within a web page, related to mouse input. This includes the absolute (more specifically, window-relative) mouse position and identifying the elements under the mouse pointer. Both clicks and hovering over elements need to be logged. If the user is familiar with the task, the recorded data must show the intentional and straight mouse movements.
- Other user actions, such as scrolling, window resizing and any keys pressed.

In order for the proxy to work with JavaScript-powered web applications, special care must be taken to prevent the proxy from interfering with any JavaScript already used by a website.

4.2 Implementation

UsaProxy adds JavaScript to every requested web page on the fly when delivering the page. The monitored data is periodically sent back to the proxy and stored in a log file.

4.2.1 Processing HTTP Requests and Responses

The application works as follows: Every client request is passed on to the respective web server. The server response is first checked for the Content-Type text/html. Figure 2 shows the two possible alternatives: In case any other content type such as image/gif is identified, the response is simply forwarded to the client. If HTML is detected, the following is added in the HTML's head element to include the monitoring JavaScript:

```
<script src='http://lo.lo/proxyscript.js' type='text/javascript'>
```

Additionally, the Content-Length header's value is increased by the number of bytes that were added. The addition of the above line is the only change made to the HTML.

Some servers do not send plain text/html content, but compress their response and use a Content-Encoding: gzip header. Due to the fact that adding the monitoring HTML in compressed streams would be very difficult, data compression is simply suppressed by overriding the client's Accept-Encoding header and using a value of "identity" instead. If a web server receives a request marked with that value, data will always be sent uncompressed.

4.2.2 JavaScript for Client-side Usage Tracking

The UsaProxy JavaScript is implemented in a way that does not interfere with any JavaScript already being used by a website. Generally, event handlers such as onclick are used to capture and handle user actions. Unfortunately, with the traditional registration model an event handler may only be registered exactly once for a single element. This becomes a problem when an event handler must be attached, but another handler is already registered for the same event. For instance, if a new onclick handler is defined, it will overwrite an old one. In the same way, the UsaProxy event handler might be overwritten by later JavaScript code which is loaded by the HTML page.

To address this problem, two advanced event registration models were added to browsers: The W3C model works with Netscape 6 and Konqueror/Safari, the Microsoft model with Internet Explorer 5+. Both models work in Opera 7. With the new models, multiple event handling functions can be added to elements without problems. With the W3C model, handlers are attached using the addEventListener() method whereas Microsoft's model uses the attachEvent() method. By using these models, it is possible to invoke the UsaProxy-specific monitoring functions without influencing the page's event handlers.

Events are objects with properties. Examples of properties which are of interest for user activity tracking are the target element of a click, hover events or the current mouse position. Since the UsaProxy script is not aware of what elements are used in the document and what names or IDs have been assigned, the event handlers are defined for the root element of the page which is the document element. The Document Object Model (DOM) of every browser gives access to all lower-level elements of HTML documents and their properties such as name, ID, href, src and so on. Events triggered on a certain DOM level are usually forwarded to the root element. Any click on a button or link will automatically "bubble" to the document element. This way, every lower-level event can be recognised and the respective event properties are available for capturing at the root level without having to touch the existing code. This results in the following data that may be monitored by the UsaProxy script together with a timestamp and the user's IP:

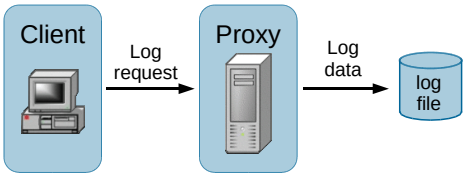

Figure 3: Log data is sent to the proxy by the JavaScript running on the client.

- The page load and resize events together with the browser window's width and height
- Page focus, blur and unload events
- Mouse click and hover events together with the target element identified by its ID, name, href or src property and the mouse position coordinates measured from the top left corner of the document
- Mouse movements together with the mouse position coordinates
- Scroll events together with the vertical position of the scrollbar
- Key press events, including the key that was pressed

Scrolling and mouse movements cannot always be recorded when the respective event is triggered because every slight touch of the scroll bar or every mouse movement would be recognised. Capturing every change and sending it to the proxy for logging would not only lead to significant network overhead and a bloated log file, but also to a tremendous overhead on the client side. For this reason, a periodical scroll and mouse movement capture is used by the proxy. A JavaScript function is called at regular intervals. It only logs a new value for the position of the scrollbar (respectively mouse) if the position has changed.

4.2.3 Logging the Tracked Data

Once the JavaScript running on the client side has collected the captured user data, it needs to be transmitted to the machine which acts as the HTTP proxy. This is done by instantiating a JavaScript Image object and setting its source to the address http://lo.lo/img.jpg. Additionally, the data to be logged is appended as a parameter. When proxying HTTP requests, UsaProxy checks for the special site http://lo.lo, captures the log request and stores it in the log file as outlined in figure 3.

However, not only the client-side information is tracked. In order to correlate usage data with the web server's actual reaction to client requests, the server responses (both HTTP headers and text/html content) are also captured and stored on the proxy. Additionally, as shown in figure 2 a log entry

is composed for identifying the file the captured response was stored in. This way, the exact HTML code output by dynamic websites is available for later inspection. Furthermore, server instructions such as redirects to another location can be identified and merged with the user inputs that evoked the server reaction.

4.3 Data Visualisation

Possible visualisations of the aggregated UsaProxy usage data range from ordinary listings of web metrics and website statistics to complex screenshot annotations. Some of these are also mentioned in section 6. The design of the proxy allows for the following scenarios:

- Accurate traffic reports that include listings of all visitors, their page visits (hits and views), visit entry points, the most common navigational paths through a website, window width, most common browsers and platforms with additional usability-related data such as the average time spent on the page before a link was clicked, scrolling activities, window resizing and average window size, popular mouse positions, the most frequently hovered-over or clicked links/buttons, and so on.

- One can visualise the visited web pages and paths through a site using special diagrams, similar to the visualisation module of the WebQuilt application [4]. Such a diagram can use thumbnail pictures of the pages with the possibility to enlarge them and connect them with arrows. For instance, emphasised arrows might indicate more heavily traversed paths or the optimal, designer-defined path. This visualisation might be combined with a number of symbols representing e.g. the time that was spent on the respective page, or buttons that were pushed.

- Mouse movements can be visualised by lines ranging from simple, edged constructions to fluid, dynamic paintings. The representation can vary depending on speed or direction of movement, e.g. by assigning the line width according to the velocity.

- Overlay of the usage data and the web metrics on the currently evaluated web page is imaginable either using a screenshot or directly within the browser. For example, the popularity of links and buttons might be displayed in a dedicated part of the screen together with other statistical information. Mouse movements could also be overlayed on the web page – figure 4 shows a simple example.

- As noted previously, a replay of the actions of a certain user on a web page appears feasible, either a live replay (e.g. for customer support) or a replay which happens later, for example while interpreting the results of a user test.

5. CASE STUDY: A USER TEST

To analyse the usefulness of the HTTP proxy approach, we conducted a small user test with our implementation. For this, 12 test participants used a computer whose web browser had been reconfigured to use the usability proxy instead of connecting to websites directly. The users were

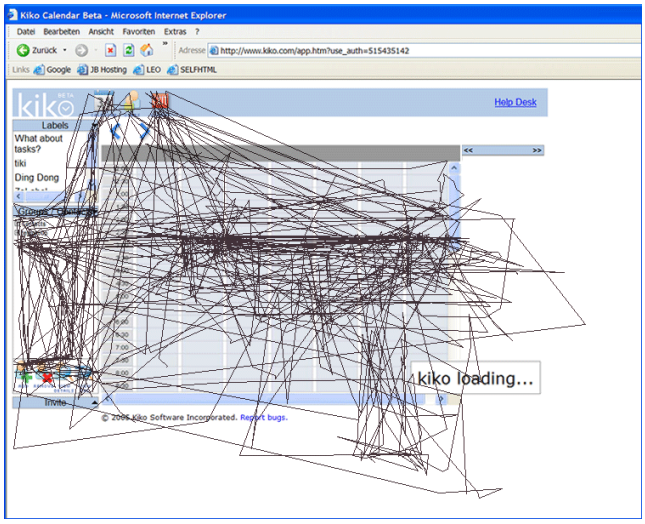

Figure 4: Mouse trails recorded by the HTTP proxy, combined with a screenshot of the website.

told to perform tasks on two different websites. Due to the test subjects' background (10 students of media informatics, 2 members of the media informatics staff), they can be regarded as experienced web users. The proxy itself was running on a nearby computer to modify the web pages and to log user actions.

The test users were presented with two tasks which are difficult to track with some of the other available user action tracking approaches (see section 6):

- On `http://www.wikipedia.de`, an online encyclopedia, the task was: *Find the FAQ entry on how to insert an image on a Wikipedia page.*
  In order to accomplish this task, a significant amount of scrolling is necessary. Additionally, navigating to the required information is possible in a number of different ways, such as filling in a search form or using the browser's built-in text search on a large page.

- For the website `http://www.kiko.com`, which offers an online calendar, the users were told: *Set up the user group "Testers" and an appointment titled "usability test" which should take place on Tuesday from 11 to 12. The Kiko application will automatically create a demo user in that group, who should be the only participant of the appointment.*
  The Kiko service is a JavaScript application. For the main calendar, the Kiko server does not send a new HTML page whenever there is a state change (e.g. a new appointment has been created). Instead, JavaScript and CSS are used to change the page at the client side.

Example proxy log Figure 5 shows a short excerpt from the log that the proxy produced during the test. Only a small selection of lines from the log are shown to illustrate the different log events. The events in the figure include mousemove (pointer position changed), mouseover (the pointer was moved over a div HTML element or similar), focus (the cursor moved into an input field) and others. The

```
141.84.8.77 2005-10-25,11:5:57 http://www.kiko.com/ serverdata 12
141.84.8.77 2005-10-25,11:5:58 http://www.kiko.com/ load width=1280;height=867
141.84.8.77 2005-10-25,11:6:2 http://www.kiko.com/ mousemove x=672;y=7
141.84.8.77 2005-10-25,11:6:2 http://www.kiko.com/ mouseover x=731;y=457 target=link:http://www.kiko.com/contact.htm+linktext:Contact
141.84.8.77 2005-10-25,11:6:6 http://www.kiko.com/ click x=815;y=231 target=id:SPAN16
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mousemove x=849;y=352
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mouseover x=472;y=296 target=id:DIV144
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mouseover x=161;y=229 target=id:left_bar
141.84.8.77 2005-10-25,11:6:38 http://www.kiko.com/app.htm?use_auth=678397351 click x=147;y=183 target=unknown:scrollbar
141.84.8.77 2005-10-25,11:6:40 http://www.kiko.com/app.htm?use_auth=678397351 mousemove x=148;y=138
141.84.8.77 2005-10-25,11:6:50 http://www.kiko.com/app.htm?use_auth=678397351 click x=26;y=507 target=id:IMG14
141.84.8.77 2005-10-25,11:6:50 http://www.kiko.com/app.htm?use_auth=678397351 focus
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=T
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=e
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=s
141.84.8.77 2005-10-25,11:47:45 http://de.wikipedia.org/wiki/Hauptseite scrolledTo y=399
```

Figure 5: Small sample of the log output produced by our HTTP usability proxy. The output includes mouse movements, keypresses and scrolling. Mouse coordinates are mapped to buttons and links on the page.

serverdata line is followed by an ID which can be used to retrieve the HTTP headers and content sent by the server, which is stored by the proxy for later inspection.

Test results Due to the detailed logging, it is easy to extract data about the test from the log. For example, a visit to specific pages marks the start and end of each task. It is also possible to determine if the user visited a certain *area* of a page, either by moving the mouse pointer over it or by scrolling to a certain position.

This way, it was no problem to determine the average time taken to complete the Wikipedia task (1:47 minutes; 1 out of the 12 participants failed to complete the task). An analysis of the different navigation paths shows that only 4 users took the optimal path. For the 6 users who used Wikipedia's search facility, the exact search strings are recorded in the logs. The proxy also continued to track one user who temporarily left `wikipedia.org` and used Google search to find the desired page.

Additionally, the proxy proved very useful in determining how exactly users completed the task. For example, the logs show that 3 users pressed Ctrl+F and typed in a search query using their browser's built-in search facility. This appears in the logs as a keypress of "F", followed by just a single scroll event, whereas users who scrolled manually through the page created several scroll events.

When performing user action logging for Kiko's calendar application, it became clear that the logs are sufficient to determine the users' actions despite the fact that the calendar is a highly dynamic JavaScript application: For the logged events like mouseover and click, the part of the JavaScript-generated GUI can be determined easily via the id attribute of HTML elements, even for things like Kiko's context menu which only opens when the right mouse button is clicked over the calendar. For instance, one user created an appointment on Monday instead of Tuesday, which is obvious due to a click entry with a target of id:selectionlayer7_1 (instead of id:selectionlayer7_2). Figure 4 shows a visualisation of all the mousemove data obtained during the user test.

Furthermore, even though Kiko uses JavaScript extensively, the JavaScript code of the proxy did not interfere in any way with the code of the JavaScript application.

6. RELATED WORK

6.1 Approaches for User Tracking

A number of previous publications discuss solutions to problems which are relevant to our work. This includes the task of tracking user actions on web pages using client-side scripting, and the development of solutions which map tracking data to the GUI elements on a web page.

Using an HTTP proxy for tracking In [4], a proxy concept is used to log the pages visited by users on the web. In contrast to our solution, no client-side tracking takes place, so only information about visited URLs is available to the proxy. The focus of the work is on visualisation techniques for the recorded log data.

Mouse tracking using client-side scripting In [5], Mueller and Lockerd introduce their idea to use embedded scripting to track mouse movement (position and associated timestamps) and to send the logged data to a server for later analysis. Due to its being an extended abstract, the paper is low on details, but it appears that the approach is restricted to the logging of mouse coordinates (i.e. no scrolling, keypresses etc) and that individual objects on the web pages (such as buttons) are not identified by the client-side code. HTML pages appear to have been prepared manually for the user study by inserting scripting commands.

Our HTTP proxy implementation uses this approach to record user actions. However, our solution supports more detailed tracking (e.g. of window scroll events) and has been carefully written not to interfere with any JavaScript already present on the web page.

For their WebVCR system [1], Anupam et al. also use JavaScript (together with a Java applet and LiveConnect) to track user actions for the purpose of implementing "smart bookmarks" to web pages which are not reachable via a normal URL, e.g. because submission of a form is necessary. While this tracking does not include mouse movements, it would be possible to add support for logging of mouse-related information.

Unfortunately, the client-based logging approach of WebVCR suffers from the following problem: Since its implementation in the year 2000, the JavaScript security model of all major browsers has been modified to be much more

restrictive, and the system does no longer work with current browsers using their default security settings. This is due to the fact that the WebVCR applet needs to be able to make "cross-domain" modifications to pages fetched from arbitrary servers.

Our own logging approach avoids the problems with browsers' security models by modifying pages on their way through the HTTP proxy.

Client-side tracking software The work by Goecks and Shavlik [3] allows tracking of user actions for Microsoft's Internet Explorer (MSIE) by means of a program which needs to be installed on the user's computer. The level of detail of the logs is quite coarse due to limitations of the implementation: For each page that is visited, only the number of clicks and the amount of scrolling and mouse activity are recorded. However, this is sufficient for the paper's goal of measuring the user's level of interest in the page.

Client-side tracking software in addition to eye tracking In [6], two different tools for user tracking on web pages are introduced. The *WebLogger* application tracks users' actions on web pages, such as navigating between pages and scrolling. Additionally, it records data from an eye tracking device. It is not clear whether tracking of mouse movement and keypresses is supported, but this seems likely. The tracking is achieved with a program which is installed on the machine. As the browser to use, only Internet Explorer is supported. *WebEyeMapper* is later used to analyse the log data, which among other things involves mapping window coordinates to links, buttons etc. on web pages. It also allows an exact playback of the test user's browsing session.

The work from [6] is similar to our own approach. The two tools allow for detailed logging including eye tracking data. However, they are more restricted with regard to the platform on which tests can be conducted (Windows with MSIE), and with regard to the scenarios of their use – for example, it is not possible to invite arbitrary users from the Internet to take part in a user study. Furthermore, an additional post-processing step is necessary to obtain useful data, such as *which* button was clicked rather than that there was a click at certain coordinates. This makes it difficult to use the tools in situations where real-time access to the data is necessary.

Server-side tracking For WebVCR, a server-based approach to logging is described as an alternative to the client-based solution that was mentioned above. It works by rewriting all URLs inside web pages to point to the server which hosts the tracking server. No implementation is presented in [1]. It would probably be difficult to make the approach work with web pages which create URLs dynamically using JavaScript or which use AJAX technology such as the XMLHttpRequest object. Furthermore, the system would stop tracking a user once the user entered an address manually, such as the person who used Google search in the case study from section 5.

6.2 Mouse Movement and Eye Movement

Our approach to user activity tracking does not include eye tracking. This could be regarded as a drawback because eye tracking can provide detailed information about how users scan and read web pages. On the other hand, eye tracking technology is expensive and requires the test user to leave their usual web browsing environment in order to use a computer with eye tracking capabilities. In the light of these arguments, the possibilities of deducing the user's gaze direction in the absence of eye tracking have been the topic of previous work.

Correlation between eye and mouse movements How closely related are the position of the mouse pointer and the position of the user's gaze on the screen? According to [2], predictions about the probable direction of the user's gaze can be made in a number of circumstances. For instance, in the experiment that was conducted, if a region on a web page was visited by the mouse pointer, there was a 84% chance that it was also visited by the user's gaze. Similarly, in the case of sudden mouse movements within or between regions of the page (saccade), the user's gaze was inside the involved region(s) in over 70% of all cases. The figure of 70% only applies if the destination region of the mouse movement contains content, i.e. is not a blank or ornamental part of the page.

Related to this, the authors of [5] observed that users would sometimes move the mouse pointer over an empty part of the web page. The cited reason for this is that a user did not want to click on a link accidentally while reading the page. During the user study, it also became clear that the mouse pointer is often used as a reading aid when scanning through menus on the web page.

Making users move the pointer using font/colour changes An interesting approach to the problem of correlating pointer and gaze position is taken in [11]: By default, the text on the page is illegible, and the user needs to move the mouse pointer over an area in order to read its contents. To prevent it from being read, the text can either be covered with a black bar, the text colour can be changed to be almost the same as the background colour, or the font size can be set to a very small value. The formatting changes are implemented using browser-independent JavaScript.

The paper only discusses the implementation of this "poor man's eye tracking" approach in an e-learning system. Unfortunately, it does not include a study to examine how much the changes to the displayed page impair the user's browsing experience, and how much this could influence his behaviour. For example, the missing text will prevent the user from making a quick scan of a page upon first seeing it. Furthermore, a delay of 0.7 seconds is suggested – only after this time, the text below the mouse pointer becomes legible. This might make the user impatient, or might even annoy him.

Making users move the pointer using blurring The *Enhanced Restricted Focus Viewer* [10] uses a similar concept as the e-learning system mentioned above. However, text is not made illegible by means of colour/font changes, but by blurring the entire page except for a quadratic area around the pointer. Again, it is not clear whether this will impair the user too much. The introduced tool is designed to be used with arbitrary web pages. However, these pages must be prepared manually for a user study, as the program only works at the level of bitmaps and needs to be told about clickable "link" areas. Because the pages are shown to the

user as bitmaps, websites with highly dynamic, JavaScript-generated content cannot be examined. Also, the mapping of pointer coordinates to page areas is only possible if the areas' coordinates are made known to the program manually.

7. CONCLUSION

In this paper, detailed tracking of user interaction on web pages was discussed from several perspectives. Going beyond the usual application of tracking technologies for user tests, we have looked at a large number of possible fields of use, ranging from inviting Internet users for usability tests, self-adapting websites, enhanced customer support through real-time tracking and user profile creation to advanced visualisation techniques for usage data.

We have introduced a sophisticated, yet flexible and transparent solution for user activity tracking. Compared to existing approaches, it allows user tracking for web applications which make heavy use of JavaScript, does not require manual preparation of web pages for tracking, and does not require control over the client or server machine. It will also help to lower the cost of usability evaluation. Furthermore, its extensible architecture allows additional functionality to be added – for example, it would be possible to add JavaScript which forces the user to move the mouse where they are looking, similar to [11] (see also section 6.2).

This work raises privacy concerns: Using our technology, the actions of users on web pages can be observed with an accuracy which is unprecedented for tracking solutions without client-side installation of software. If our approach is abused, this can happen without the users' knowledge. It is the responsibility of anyone performing user tracking to inform the subjects of the tracking about its use. At a minimum, we recommend that a user be asked for his consent before any logging takes place. Furthermore, if arbitrary visitors of a website agree to their actions being logged, their consent should only be considered valid for a few hours, or until the end of their browser session.

Acknowledgement This work was funded by the BMBF (intermedia project) and by the DFG (Embedded Interaction Research Group).

8. REFERENCES

[1] V. Anupam, J. Freire, B. Kumar, D. Lieuwen: Automating Web Navigation with the WebVCR. In *Proceedings of the 9th International World Wide Web Conference WWW9*, Amsterdam, Netherlands, May 2000

[2] Mon-Chu Chen, John R. Anderson, Myeong-Ho Sohn: What can a mouse cursor tell us more? Correlation of eye/mouse movements on web browsing. In *Proceedings of the Conference on Human Factors in Computing Systems CHI 2001*, extended abstracts on Human factors in computing systems, Seattle, Washington, USA, April 2001

[3] J. Goecks, J. Shavlik: Learning Users' Interests by Unobtrusively Observing Their Normal Behavior. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pages 129–132, New Orleans, Louisiana, USA, 2000

[4] J. I. Hong, J. Heer, S. Waterson, J. A. Landay: WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. In *ACM Transactions on Information Systems (TOIS)*, Volume 19, Issue 3 (July 2001), ISSN:1046-8188, pages 263–285

[5] F. Mueller, A. Lockerd: Cheese: Tracking Mouse Movement Activity on Websites, a Tool for User Modeling. In *Proceedings of the Conference on Human Factors in Computing Systems CHI 2001*, extended abstracts on Human factors in computing systems, Seattle, Washington, USA, April 2001

[6] R. W. Reeder, P. Pirolli, S. K. Card: WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-use Studies. In *Proceedings of the Conference on Human Factors in Computing Systems CHI 2001*, extended abstracts on Human factors in computing systems, Seattle, Washington, USA, April 2001

[7] D. Salber, A. K. Dey, G. D. Abowd: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99)*, Pittsburgh, PA, May 15–20, 1999, pages 434–441.

[8] A. Schmidt: Implicit Human Computer Interaction Through Context. *Personal Technologies*, Volume 4(2&3), June 2000, pages 191–199

[9] A. Schmidt, M. Beigl, H. W. Gellersen: There is more to context than location. *Computers & Graphics Journal*, Elsevier, Volume 23, No. 6, December 1999, pages 893–902.

[10] P. Tarasewich, S. Fillion: Discount Eye Tracking: The Enhanced Restricted Focus Viewer. In *Proceedings of the Americas Conference on Information Systems AMCIS 2004*, New York, NY, USA, August 2004

[11] C. Ullrich, E. Melis: The Poor Man's Eyetracker Tool of ActiveMath. In *Proceedings of the World Conference on E-Learning in Corporate Government Healthcare and Higher Education eLearn-2002*, pages 2313–2316, Montreal, Canada, 2002