Clever Tracking User Behaviour over the Web: Enabling Researchers to Respect the User

Evdokiya D. Ignatova[1]
[1]Brunel University
Uxbridge, Middlesex
UB8 3PH, UK
ev_ignatova@yahoo.com,
willem.brinkman@brunel.ac.uk

Willem-Paul Brinkman[1,2]
[2]Delft University of Technology
Mekelweg 4, 2628 CD Delft
The Netherlands
w.p.brinkman@tudelft.nl

ABSTRACT

Concerns over automatically tracking users' actions while respecting consent, privacy and users' rights motivated the development of CleverTracker. CleverTracker is a remote action-tracking software framework, which researchers can use to collect data about users' interactions with applications while respecting ethical issues. Users are in control of the recording process (through start and stop functionality), can opt out from it and can view the collected data. The open source framework is designed to support desktop, web application and multiple programming languages.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces– *evaluation/methodology.*

General Terms

Measurement, Experimentation, and Human Factors.

Keywords

Computer-assisted usage analysis, remote tracking, action recording, logfile analysis, and research ethics.

1. INTRODUCTION

One way to examine the usability of a system is to test it with real users and analyse collected usage data. There are different approaches in collecting usage data. Traditional evaluation methods include questionnaires, interviews and observations. This approach may have a limited scope as employing users may be expensive, and the experimental conditions of the setting may cause users' interactions to be different from what they would be in real life [1]. Today's advances in broadband Internet services provide ways for applications to record user interaction unobtrusively over long periods of time. This can be done on a larger scale and independent from user locations. Researches may benefit greatly from tracking user actions remotely; however, this method has a number of recurrent issues, which do not seem to have been fully addressed.

A major problem in collecting users' data is on respecting users' privacy and ensuring an ethical approach to the data collection process. As Tang et. al. [5] point out, even though remote tracking may be unobtrusive, it is still invasive and as such users should be in control of the recording process. This idea agrees with the outcome of the afternoon discussion at the Interaction Tracking workshop [1] in 2006. Delegates recognised that some users have reservations to give consent for recording their data, because they might perhaps not know or understand what they are getting themselves into. Several suggestions were put forward to overcome this, such as allowing users to opt-out and view their recorded data. A number of research teams have attempted to create their own tracking frameworks, and have adopted different approaches to respect users' privacy. For example, the GRUMPS framework [3] records low-level keystroke events; hence privacy and ethical issues have been a focus of concern as users might enter their password or use their browser for online banking. GRUMPS therefore initially obfuscated all key presses. However this limited the analysis possibility of the data, and therefore it was later changed to obfuscating only alphanumeric, numeric and special character data. Looking at the framework from users' point of view it did not seem to provide visual means to control the recording process. A different approach can be seen in the PROSKIN project [2], which attempts to solve some of these issues. PROSKIN makes use of a custom-built tracking component which collects interaction data from an internet radio application over the web. Users can opt out and view what data has been recorded; still the implementation of these functions is limited, as opting out means uninstalling the entire application and viewing the recorded data involves reading text files. Currently there also seems to be a lack of any unified user tracking framework, which may be extended to any platform, programming language or database. Most tracking frameworks seem to be either application-specific or limited to a few programming languages or a single platform. This means that researchers and software developers are forced to spend each time efforts to create a log mechanism rather than focusing on the analysis of the data.

CleverTracker is a remote tracking framework put forward in this paper, which attempts to solve the issues discussed above. It enables researchers to respect users' rights. It aims to achieve this by providing visual controls, which allow users to start, stop and pause data recording, as well as view logged data and opt out from an online research study. CleverTracker is also cross-platform compatible, which provides further flexibility to researchers to choose their preferred operating system, programming language and relational database. To encourage its use, reviews and the further extension of the software, the

Published by the British Computer Society
Volume 2 Proceedings of the 21st BCS HCI Group Conference
HCI 2007, 3-7 September 2007, Lancaster University, UK
Devina Ramduny-Ellis & Dorothy Rachovides (Editors)

CleverTracker software framework is distributed as an open source project hosted on SourceForge[1], under the BSD license.

2. FRAMEWORK DESIGN

The framework is based on the client-server model. The server component stores data permanently to a relational database for later analysis. It is written in Java and can be configured to support any database with a JDBC driver. This allows the server component to be set up and run on any operating system. The client component is a library which is used in the code of the application under evaluation. As such there are multiple client libraries to match the various programming languages that applications may be written in. Currently, these include Java (for desktop applications) and JavaScript (for web applications). The interaction data is transferred from the client libraries to the server component in the form of HTTP messages, making it easy to extend the framework to support additional technologies in the future. Each message represents a single event generated from the application under evaluation and is sent by the client with the fields listed in Table 1. The "messageId" field acts as the unique identifier for each message. All unique identifiers are generated locally on the client side, with the help of UUID generators. This makes it possible to use the framework in partially connected environments where internet access may not always be available.

Table 1. Description of the parameters of the log method.

Parameter	Description
messageId	The messageId serves as a unique identifier for messages sent from any client to the server and logged in the central repository.
sessionId	The sessionId stands for a single run of the application providing the interaction data.
userId	The userId anonymously identifies a user of the software application.
messageType	The messageType provides an extra categorisation of the generated messages.
eventOrigin	The attributes eventOrigin describes the source in the application that triggered the message generation, eg: `brunel.Converter.buttonClicked`
eventMessage	A free text field that can contain any data which is relevant to the researcher.
timestamp	A record of when the message was passed from the software application to CleverTracker client.

The "userId" field is also automatically generated by the client libraries as a sequence of characters which cannot be directly traced back to the users' details. This field then allows aggregating and analysing usage data for individual users, while respecting their anonymity. Unlike the "userId" field which remains the same, the "sessionId" field is generated for each run of the application. It can be used to identify for how long people use an application, or what functionality is used during each session. The "messageType" field can be set in advance to provide some categorisation of messages, for example an "INFO" message or "BUTTONCLICK" message. This field is free text. The actual information which is recorded is passed as free text using the "eventMessage" field. This allows any type of textual data to be passed in the form of a string. The eventOrigin allows locating the source of the message. It is a hierarchical classification which identifies the user interface element which triggered the message. In Java desktop applications this is automatically captured as the package name, class and method name which generated the event.

2.1 Message Flow

The intended message flow (Figure 1) between the clients and the server for this framework is the following: A user interacts with an application, through some input device, such as keyboard or mouse. The application then responds to this input by executing some method or procedure. Additional code should be inserted at this point which interacts with the CleverTracker library and captures the event. The library then checks whether the user has allowed the tracking of data and sends the events to the server component. If the server is not reachable the messages are queued locally and resent later once the server is available. When the server receives the message it is stored in a database. The server then sends a response to the client to confirm the storage of the message. A researcher may then access the database directly and perform detailed analysis of interaction data using their chosen technique. Such analysis may be achieved, for example, using data mining tools or custom defined SQL queries. To provide a degree of fault tolerance, the client is designed to continue sending the same message to the server, until it receives an acknowledgement, thus compensating for loss of messages. The server checks the "messageId" field of every incoming message and only stores each message once, even if it is received multiple times.

3. USING CLEVERTRACKER

3.1 Server Component

The server component runs as a J2EE application, and is designed to be compatible with most Servlet containers (such as Apache Tomcat or Jetty). The database connectivity can be set up by editing an XML configuration file on the server. This file contains the individual SQL statements that are used to insert and select data, and as such the server component may be configured to work with any database engine which provides a Java driver.

3.2 Client Libraries

An application that wants to make use of CleverTracker has to be customized by a software developer to call the programming interface of the appropriate client library. The following code snippet is a sample method called "buttonClicked" from a currency converting application (Figure 4). This application is written in Java and hence makes use of the Java CleverTracker client library. This code illustrates a call to the programming interface of the library to log the amount money and the currency type which a user wishes to convert.

```
private void buttonClicked() {
1. double amount = getAmount();
2. String currency = getCurrency();
3. double converted =
      exchange(amount, currency);
4. recorder.logMessage("User converted " +
      amount + currency, "INFO");
}
```

[1] For the source code and documentation see http://clevertracker.sourceforge.net/

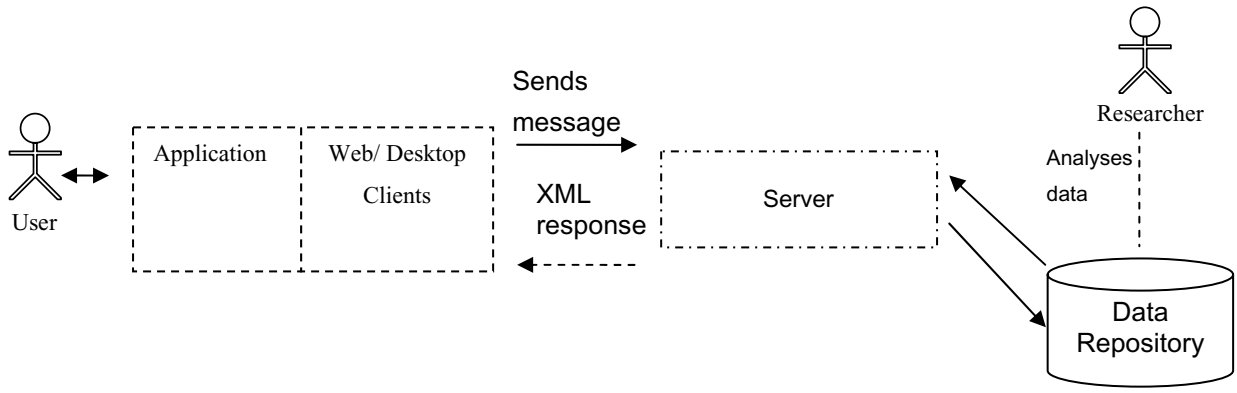

Figure 1. Message Flow.

The client library is called in line 4 and two pieces of information are passed separated with a comma. The first part is the "eventMessage" field, while the second part is the "messageType" field. Once this information is passed to the client library and it has been confirmed that the user has allowed data collection, the rest of the fields are automatically generated. Then the message is sent to the server. The sending of the messages is processed separately while the application is running, and as such including this code does not produce any noticeable delay. The presence of this additional code is transparent to the user. A sample logged message in the database record is illustrated in Figure 2.

messageId = 05135554-a343-4de3-b017-44121e129823-1
messageType = INFO
userId = f795b2e5-95e3-4eda-adc5-0cbbaca1090e
sessionId = 05135554-a343-4de3-b017-44121e129823
eventMessage = User converted 10USD
eventOrigin = clevertracker.DriverGui.buttonClicked
timestamp = 2007-05-24 3:42:59.000000921

Figure 2. Example of a recording.

4. VISUAL CONTROLS

CleverTracker provides additional user interface elements, which allow users to be in control of the recording process. When the application makes its first attempt to log data, the user is prompted that the application is going to collect data and asks for their consent. If the user is happy to allow this action, data collection begins and visual controls are provided as a small system tray icon with a right-click menu (Figure 3). This menu allows the user to monitor the status of the data collection process, which could be either "Started" or "Paused". By clicking on the "View Recorded Data" button, a sample of the logged data from the current session is displayed to the user. Another important menu option is "Why is data being recorded?". It allows the researcher to provide further information on what type of data is being collected as well as external web resources, where the user can read more about the study. This menu button for example could also lead to a discussion forum, where the user can post questions and communicate with the researcher directly.

The "Start Recording" and "Pause Recording" options allow users to pause the data collection for a single session and restart it at a later stage. On the other hand, the "Stop and Exit" button stops the recording functionality for an entire session. The "Opt-out" option permanently disables the CleverTracker functionality; hence the user can choose to leave the study.

Figure 3.User Visual Control Menu.

5. USABILITY EVALUATION

An initial usability evaluation of the framework was conducted, which aimed to evaluate clarity and usability of the user controls provided by the CleverTracker client libraries. In order to evaluate the visual interface of the clients, a series of informal semi-structured interviews were conducted. The interview questions were centred on the ethical requirements, which were the focus of the CleverTracker framework. Five participants took part in the study. They were all undergraduate students in their last year of their Computer Science degree and they were between 20 and 26 years old.

After given a short introduction on the tasks which they need to fulfil, the participants had to use both a web and desktop currency converter application (Figure 4). The complexity of the application was relatively simple, as participants had to enter a value, select a currency, and press the convert button to get the value in Euro. After completing their task, participants where interview in a short debriefing session. The interview questions covered key issues such as the alerting system, the clarity of information provided and the usability of the visual controls.

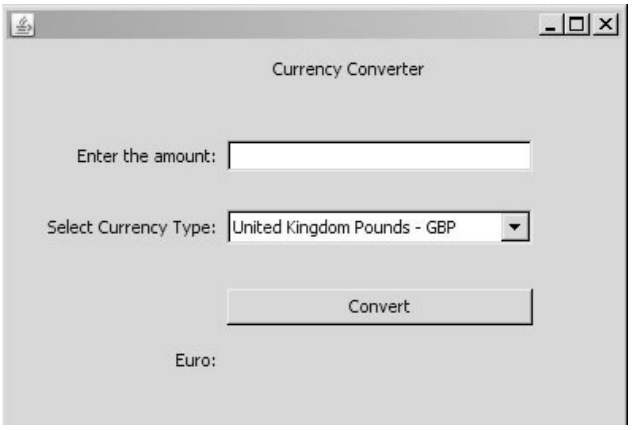

Figure 4. Currency converter application.

A number of interesting observations were made. All of the participants seemed confident in using the "Start Recording", "Pause Recording", "View Recorded Data" and "Opt-out" options. Almost all participants realised immediately that data was being collected from the two applications, which meant that the alerting system seemed to work as expected. Some of the participants liked the fact that the alert pop-up comes when they actually used the functionality of the web application. In their opinion this helped them distinguish the data collection alert from other spam message which according to them were frequently seen, while browsing the web. In the desktop application, the alerting pop-up with the opt-out feature seemed also clear, however some of the users did not seem to notice immediately the system tray icon for the visual menu, even though there was an information pop-up indicating that there was a menu there. A potential way to overcome this problem in future editions of this platform might be to convert the system tray menu into a floating window appearing next to the application or even it could be part of the application window itself.

When asked whether they would feel comfortable participating in studies that was tracking data about their use of a web or desktop applications, most participants were rather positive about it. Although a few mentioned that they seemed more at ease if they were participating in a web based study instead of their desktop machines. Overall, the evaluation study showed encouraging results. A further step would be organising a similar more hands-on session with developers and researchers to gain an understanding of what features they believe need further improvement.

6. LIMITATIONS AND DISCUSION

One limitation of the JavaScript CleverTracker client became apparent when the client library was tested with a web browser, which had its pop-up blocker on. In that case the CleverTracker JavaScript alert asking users for their consent to collect data was blocked, which resulted into having the collection process paused for the entire user session. Another important issue, which has not been yet addressed, is the collection of sensitive data with the framework. This could be an important issue if the application under evaluation stores passwords or credit card details. Currently, CleverTracker does not apply any filtering on what data is collected from the user. This has been left to the discretion of the researchers. They decide what data is actually recorded. Still, with the View Recorded Data option, users will be in the position to see what type of data is recorded, making the recording process transparent. This awareness might help them to understand when to use the Start Recording and Pause Recording functionally when participating in study which records their actions.

7. FINAL REMARKS

We believe that this software will allow researchers to better understand how people interact with software. Given that this is one of the first tools that spans on both web and desktop platform, it could also allow an interesting comparison between the two technologies. During every stage of the development process, it was aimed to create a reliable and extensible system, which is easy to configure and support. Naturally, taking into consideration the size of the project, there are a few areas that deserve further improvement. From functionality point of view, it would be interesting to extend further the opt-out feature. Users could be given the choice to opt out only from specific types of recording and to give them an opportunity to participate in other. Another key area of improvement might be an automatic data removal feature. This would allow the complete removal of already collected users' data from the database on request. Potential future work could also involve gathering more user feedback on how users perceive the usability of the visual controls of the framework. We equally hope to receive feedback from researchers and developers on what could be further improved in this framework. Future work could also focus on developing clients to support more programming languages, and developing software for research to extract and analyse the interaction data in data repository.

8. REFERENCES

[1] Brinkman, W.-P., HCI 2006 workshop report interaction tracking. In *Proceedings of the 2006 Workshop on Computer Assisted Recording, Pre-Processing, and Analysis of User Interaction Data.* Lulu, Morrisville, NC, 2006, 93-95.

[2] Fine, N. Personalising Interaction using Profiled User Interface Skins. In *Proceedings of HCI 2005,* vol. 2, 194-196

[3] Hilbert, D. M., and Redmiles, D. F. Large-Scale Usage Data to Inform Design. In *Proceedings of INTERACT* 2001 (Tokyo, Japan, July 9-13, 2001). IOS Press, Amsterdam, The Netherlands, 2001, 569-576.

[4] Renaud, K., and Gray, P. Making sense of low-level usage data to understand user activities. In *Proceedings of SAICSIT'04*. South African Institute for Computer Scientists and Information Technologists, Stellenbosch, Western Cape, South Africa, 2004, 115-124.

[5] Tang, J. C., Liu, S. B., Muller, M., Lin, J., and Drews, C. Unobtrusive but invasive: using screen recording to collect field data on computer-mediated interaction. In *Proceedings of CSCW '06*. ACM Press, New York, NY, 2006, 479-482.