WIDAM - WEB INTERACTION DISPLAY AND MONITORING

Hugo Gamboa and Vasco Ferreira
Escola Superior de Tecnologia de Setúbal
Campo do IPS, Estefanilha, 2914-508 Setúbal, Portugal
Email: {hgamboa, vferreira}@est.ips.pt

Key words: Human computer interaction, web, monitoring system, client-server application.

Abstract: In this paper we describe the design and implementation of a system called Web Interaction Display and Monitoring (WIDAM). We have developed a web based client-server application that offers several services: (i) real time monitoring of the user interaction to be used in synchronous playback (Synchronous Monitoring Service) (ii) real time observation by other users (Synchronous Playback Service); (iii) storage of the user interaction information in the server database (Recording Service); (iv) retrieval and playback of a stored monitored interaction (Asynchronous Playback Service).
WIDAM allows the usage of an interaction monitoring system directly over a web page, without the need of any installation, using low bandwidth comparatively to image based remote display systems.
We discuss several applications of the presented system like intelligent tutoring systems, usability analysis, system performance monitoring, synchronous or asynchronous e-learning tools.

1 Introduction

There are several types of User Interaction Monitoring systems available on the market. We classify them according to their principal usage.

Remote computer operation systems, allow the control of the user interface of a remote machine from anywhere on the Internet. Examples of these systems are: *Netmeeting* (Microsoft, 2002), a tool freely included in all windows distributions that allows, between others services, the possibility to operate a computer from a remote location (Remote Desktop Sharing); The program *Virtual Network Computing* from AT&T (Richardson et al., 1998)(Richardson and Wood, 1998), which is available for several machine architectures, enables independent operation systems interconnection. The VNC was developed in an open source effort and is freely available. The program *pcAnywhere* (Symantec, 2002) from Symantec, is the more known commercial remote desktop sharing application.

Interaction recording and display systems are capable of saving the interaction of a user, during the usage of any type of application. This type of programs are used to create training material, software simulations, online presentations among others. Examples of these programs are: ViewletBuilder (Qarbon, 2002) from Qarbon, Camtasia (Weber, 2001) from TechSmith and CamCorder from Microsoft Office.

Other interesting tools related to user interaction monitoring systems are Java Observation, Scripting and Inspection Tool (JOSIT) (Chisholm, 2001) and Widget Observation Simulation Inspection Tool (WOSIT) (Geier et al., 2001). Both are software instrumentation tool's for applications in Java and X-Windows respectively. These tools are designed to:

1. observe the user's manipulations of the target application's GUI,
2. inspect states of the GUI, and
3. initiate actions on the target application's GUI.

The X-Windows system (Scheifler, 1988) offers a protocol that enables the communication of a user interaction via network. X-Windows system is transparent to the location of the user and always uses the X-protocol to transmit and then process the user interaction in the graphics user interface of the program. The X-Windows is associated to unix and linux operating systems, and can be used to lunch remote applications. The remote user needs to have a interface emulator that understands the X-protocol. These emulators work in most of the operating systems, includ-

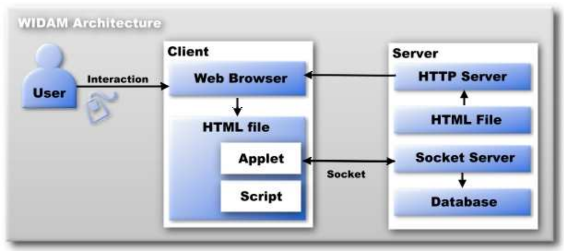

Figure 1: The WIDAM Architecture.

ing windows.

1.1 Our proposal

Our proposal, WIDAM, is a client-server application for web pages monitoring, analysis and display. The system can be called as a remote display system that enables the real-time observation of the user interaction, offering four different services :

1. Synchronous Monitoring Service — real-time monitoring of the user interaction;
2. Synchronous Display Service — real-time observation by other users;
3. Recording Service — storage of the user interaction information in the server database;
4. Playback Service — retrieval and playback of a stored monitored interaction.

WIDAM, differs from the presented systems in several aspects. WIDAM allows the usage of a interaction recording system directly over a web page, based on the Document Object Model (Hors et al., 2000) (DOM) of the web page. The system works in a normal web browser with java and javascript capabilities, without the need of any software installation. WIDAM is a light weight networked application using low bandwidth comparatively to image based remote display systems.

In the following sections we will describe in detail the WIDAM system. Section 2 mainly discusses the system architecture, presenting the server and client applications. The communication protocol developed for the WIDAM system is presented in section 3. Section 4 contains an example of the WIDAM system monitoring a game. In section 5 we discuss some possible applications of the WIDAM system. We conclude presenting future developments of the system in section 6.

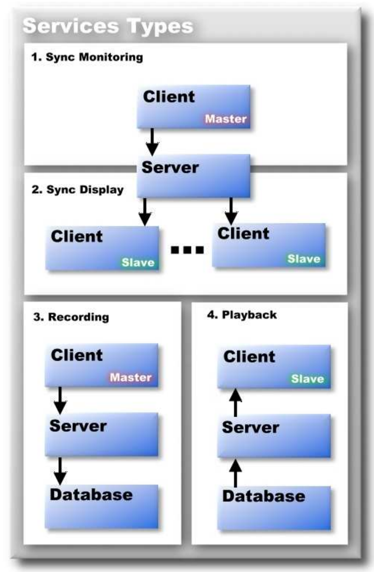

Figure 2: The services types provided by the WIDAM system

2 System Architecture

The WIDAM Architecture is composed by a client and server applications as depicted in figure 1.

The user accesses the WIDAM application via a web browser that connects to the server. Then the server sends back to the user a web page that is capable of monitoring and displaying the user interaction. This page creates a connection to the server and selects one of the services provided by WIDAM. Then the client and the server exchange messages using a specific protocol.

The WIDAM system offers four services that enable the monitoring and display of the user interaction in the web browser (see figure 2).

The Synchronous Monitoring Service is requested when the user wants to send his interaction information in real time, and he is expecting that other users will connect to the system and observe his interaction at the same time. Every time the user interacts with the browser, the server receives a notification message in order to pass the information to the clients that requested the Synchronous Display Service.

The Synchronous Display Service is requested by a user that wants to observe other user that is presently sending his interaction information via the Synchronous Monitoring Service. Every time the observed user interacts with the web page the event is sent to the server and replicated to all the clients that are connected via the Synchronous Display Service, and the interaction is simulated in the client web browser.

The Recording Service is requested when the user wants his navigation through the web page stored in the server database. The events are sent in the same way as in the Synchronous Monitoring Service, except, when the interaction information arrives to the server, this information will be kept in the server database.

The Playback Service is requested when a user wants to view a recorded monitored interaction that has been saved by a client (when this last one requested the Recording service).

When the client connects to the server it requests one of the four available services. These services are divided in two client modes: master and slave. The master controls the interaction, the slave observe the interaction. The client operates as master when he requests the WIDAM system to monitor the interaction. The Synchronous Monitoring Service and the Recording Service are the services in which the client is in the master mode. The client operates in the slave mode when he requests a service where the WIDAM will play the interaction of some other user. The Synchronous Display Service and the Playback Service are the services in which the client is in the slave mode.

2.1 The client

The client works in any web browser capable of executing Javascript code and Java Applets, independent of the operating system. When the users enters in a page of the WIDAM system, an applet is launched. This applet creates a socket connection that enables the message passing from and to the server. The client loads the html page and sends an handshaking message through the open socket, specifying which type of service is requested.

The monitored html pages the clients request to be monitored need to exist in the server. The socket connection can only be done to the server from where the java applet was loaded.

In master mode service, the script sends a request to the browser, asking for notification of the user interface events. These events are a sub set of the events from the Document Object Model Events (Pixley, 2000). In table 1 we list the events captured by the WIDAM system. Every time one of these events occur, a message is passed to the server.

In the case of a slave mode service, the web browser creates a virtual mouse pointer and waits for messages from the server specifying which event should be emulated in the web browser.

2.2 The server

The server machine is composed by two servers listening to different ports.

ID	Event handler	Event cause
0	onMouseMove	The user moves the cursor.
1	onMouseDown	The user presses a mouse button.
2	onKeyPress	The user presses a key.
3	onUnload	The user exits a document.
4	onMove	The window is moved.
5	onSelect	The user selects some text.
6	onResize	The window is resized.
7	onBlur	The window loses focus.
8	onFocus	The window receives focus.

Table 1: DOM events captured by WIDAM.

We used a http server (listening at port 80) to wait for html pages requests. We used different http servers like Internet Information System (IIS) from Microsoft or the Apache http server from Apache Software Foundation, a open source solution. The html pages needed by the WIDAM system are passed from the http server. The javascript code and applet code needed for the communication, monitoring and display operations, are also passed in the same manner.

We developed a Java server that listens for socket requests at port 8001. This socket server is always active and waiting for a request of a communication channel. When a client asks for a socket connection, the servers assigns a new socket and creates a thread to process the communications with the new client.

The client sends a service request via a handshaking message. The state diagram of the server (see figure 3) specifies the operations performed by the server in the different services types.

When the server receives a handshaking message requesting the Synchronous Monitoring Service, the server waits for data from the client and when the data arrives, the server sends it to all slaves clients that requested the interaction data. Similarly, when the server is asked for the Synchronous Display Service, it waits for the data of the monitored user (master), and when it arrives the server sends the data to the client. In the case of the Recording Service, the difference to the Synchronous Monitoring system is that the received data is stored in the database. In last service, the Playback service, the server starts retrieving the requested information from the database and creates a timer to schedule the data messages according to the intervals these messages had at the recording time, in order to correctly emulate the playback of the interaction in the client.

3 Communication Protocol

The WIDAM protocol is composed by three types of messages: Handshaking, Data and Terminate.

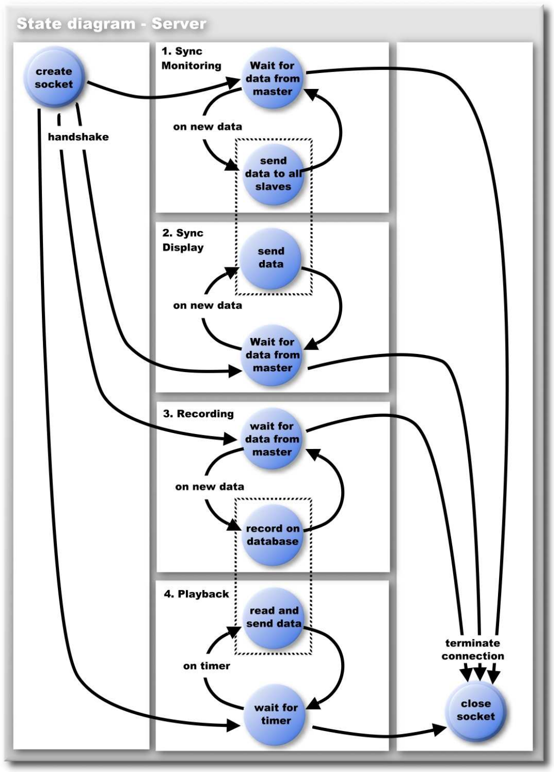

Figure 3: A state diagram of the server.

Field	Bytes
message ID	1
version	15
service type	1
entry ID	20
timestamp	4

Table 2: Handshaking message.

The Handshaking begins with the client sending a message identifying the protocol version used, the requested service type, identification information, and a timestamp. These two last items are used to select the entry in the database where the information should be recorded or retrieved. The version entry is of the form: `"WIDAM v0.2"`, indicating the version and sub version of the protocol the client will use. The timestamp entry is a value that counts the milliseconds since January 1, 1970, 00:00:00 GMT. In table 2 the structure and type of contents of the message handshaking are presented.

The Data message is sent to the server when the client is operating as master, and is sent to the client when the he is operating as slave. The message is composed by several information about the Document Object Model event generated. The event ID is the number associated to the generated event (see table 1). The document object model object ID is the number that identifies the object in the page that is connected to the event. The relative positions are the X-axis and Y-axis displacements relative to the origin of the DOM object. The absolute positions express the position of the pointer from the origin of the page. The other information field contains additional data. In the case of an `onKeyPress` event the other information contains the ASCII value of the pressed key. The time stamps indicates, with millisecond precision the time when the event occurred. Table 3 lists all the fields of the message.

Field	Bytes
message ID	1
event ID	1
DOM-Object ID	1
relative position X	2
relative position Y	2
absolute position X	2
absolute position Y	2
other information	4
timestamp	4

Table 3: Data message.

Field	Bytes
message ID	1
timestamp	4

Table 4: Terminate message.

The terminate message is sent by the client stating that it desires to close the connection. The message is composed by the message ID and the timestamp (see table 4)

The time diagram presented in figure 4 shows the exchanged messages between the client and the server in the case of the Recording Service. The client sends a handshaking message requesting the Recording Service. The client starts monitoring the web browser and sends data messages, specifying the type of event and other related information, each time a the user interacts with the web page. In the end of the navigation the client sends a termination message.

In normal operation the exchanged messages are data messages. The handshaking message and terminate message occur only once per established connection. We can approximately calculate the bandwidth required by the protocol by measuring a data message and the frequency of the exchange of messages. The messages are generated by input devices like the keyboard and the mouse. A pointing is the source of most of the messages. It sends events notification every time the mouse is moved. The pointing device is usually sampled at 50 times per second (there are some differences between operating system and types

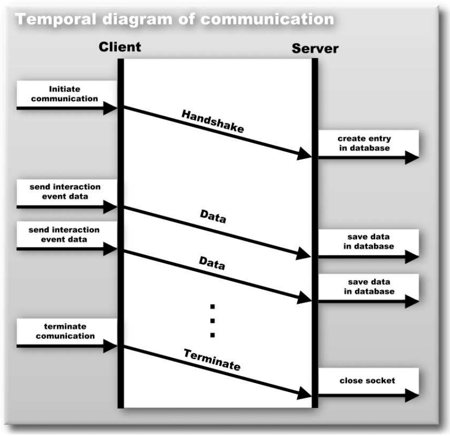

Figure 4: A temporal diagram of the protocol, in the case of the service recording.

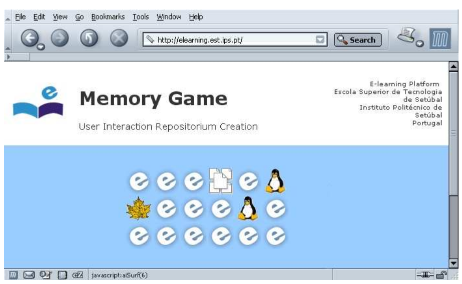

Figure 5: An example: the memory game.

of mouse but these values are very similar). We get a value of approximately 1 kbytes/s (950 bytes per second) as the maximum required bandwidth. This value occurs in the worst case of the user being always moving around the mouse (without any interruption to click or read), which is not the typical situation. Even at the maximum required bandwidth, the protocol doesn't require more than a simple Internet phone line connection.

4 Example

We used the WIDAM system in the task of monitoring the user activity while playing a game. The objective of this example is to provide a web interface to collect computer interaction data from several users in order to create a repository of computer interaction information. This repository have been used in the study of users interaction characteristics.

The reason to select a game is that games require more user interaction activity, producing more interaction events, like mouse movements, mouse clicks or key presses, per second, that normal computer interaction in web browsing, mail reading, or document producing.

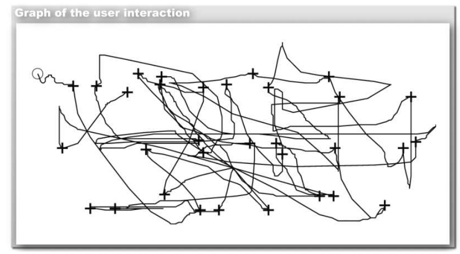

Figure 6: Graph of the user interaction in the memory game. The staring point is marked with 'o'. The pointer movements are marked along the line and the '+' mark the mouse clicks. .

The selected game is presented in figure 5 and is called the memory game. A person clicks in a hidden image that reveals itself and then tries to match the pair image that exist somewhere in the board. If the match fails, both image become hidden again.

We developed two web pages in order to use the WIDAM system. The first page is an Identification page, where the user introduces his name and a personal number. This page serves only the purpose of identification of the user, in order to several recorded games be kept with the user identification information. There is no need to an authentication system, since there are no security issues with the task of collecting user computer interaction information.

The second page is the game page that uses the entered information to call the WIDAM system, evoking the Recording service and starts monitoring the user activity while the user is playing.

Figure 6 shows a graph of a user interaction while playing an entire memory game. The graph is produced by joining every sequential mouse movement with lines and using a cross mark to indicate a mouse click.

In table 5 we show a excerpt of the recorded interaction protocol. These are data messages (see table 3 where we omitted the Other Information field since in this events it is empty.

5 Applications

The system can be applied in several tasks from education to e-commerce. We developed the system primarily as a support of an e-learning environment. A teacher opens the WIDAM system and starts browsing a set of web pages. The students open at the same time the application and observe the teacher navigation through the same set of pages. A student that could not be present at the time of the lecture can ac-

EID	DID	RX	RY	AX	AY	TS
0	8	6	20	97	215	1023049291400
0	8	5	20	96	215	1023049291460
0	7	24	20	73	215	1023049291510
0	7	17	20	66	215	1023049291570
1	7	16	20	65	215	1023049291730
0	7	16	20	65	215	1023049291840

Table 5: Example of the recorded data from a user playing the memory game. The abbreviations are defined in the WIDAM protocol: EID: Event ID; DID: DOM-Object ID; RX: Relative Position in X; RY: Relative position in Y; AX: Absolute Position in X; AY: Absolute position in Y; TS: Time stamp.

cess to a recorded interaction of the teacher a follow the steps performed during the class. Note that the system is suitable for a distant lecture or for a live lesson at a computer laboratory. Every student is at the computer and watch at the teacher movements in a web page while the teacher is verbally explaining what is doing.

Another example in the education area, specifically asynchronous e-learning, is application of the system recoding the student interaction while attending an web based lesson, for instance in an asynchronous mode, reading some texts delivered by the teacher via web. The system can support the study the lecturer attention, try to guess what areas of the lecture he is more interested or the type of contents that the lecture likes most. The analysis of the interaction could be used by an Intelligent Tutoring System in order to propose to the student the more appropriated contents.

We have already applied the system in simple monitoring of a user interaction activity as presented as example in section 4. In studying the type of interactions performed by several persons we could record their interaction for later playback and analysis of their mouse movements, key presses and other user interactions.

The system could be used in a similar way in web usability tests by identifying the difficulty of a user to find what he wants in a particular web page. The WIDAM system could record the interaction of the user and the interaction playback shows the trips of the mouse pointer until the user reaches the desired button or part of the web page.

In a costumer relationship management system, WIDAM could be used in the same mode as in a e-learning class, where a operator conduces the user thought the WebPages where he found a difficulty or wants an explanation.

In an e-commerce web site, the WIDAM system could be used in other to gain comprehension in what are the web page areas that focus the user attention, and what types of navigation is typically performed by the users.

These are several application examples that can be expanded to several areas related to the web.

6 Conclusion

In this study, we have presented a human computer interaction monitoring and display system, functioning on the World Wide Web. The system is composed of client server architecture and works in a web browser. Considering similar applications, we have found that our proposal is a new approach to this problem by being a web based application and by presenting a protocol that requires a relatively small bandwidth.

This system raises some computer ethical questions. The usage of the WIDAM system in a hidden way may be considered as a privacy violation. We consider that our main focus is applying WIDAM to education and free of this ethic questions, but as future work usage norms of the system should be created.

The project has been constructed under an open source initiative in order to freely be used by other programmers. We have opened a project account in SourceForge where the code for installing the system is available and where people interested in the growth of the project can meet to share effort in enhancing WIDAM, reporting bugs and give examples of application of the system.

We are considering some effort in improving the present version of the WIDAM system.

The protocol can be easily compressed by using a differential code in the coordinates or in the time values. Applying some compressing technique like LZW or other will further compress the resulting interaction information.

As stated before, the system can only be used in pages that exist in the server. We would like to pass this limitation by fetching the pages that the master user is interested and then inserting the monitoring code in both master and slave users.

The protocol could be extended to enable the adding of comments to interaction. These comments could be text positioned in the web page adding some information to the navigation, or the comments could be in a sound format in order to the recorded interaction be companied by a verbal explanation.

The access to the system is open in the sense that no one needs to do an identity check. So any one that knows where the system is working can start a recoding session or a playback of a recorded interaction. As future work we think that a security layer should be created with a list of users and privileges.

REFERENCES

Chisholm, M. A. (2001). JOSIT user manual. Technical Report MP 98BVSR, The MITRE Corporation, Center for Integrated Intelligence Systems, Bedford, Massachusetts.

Geier, M., Chisholm, M. A., and Cheikes, B. A. (2001). WOSIT user manual. Technical Report MTR 01B0000014, The MITRE Corporation, Center for Integrated Intelligence Systems, Bedford, Massachusetts.

Hors, A. L., Hgaret, P. L., and Wood, L. (2000). Document object model level 2 core. Technical report, W3C.

Microsoft (2002). Netmeeting. URL, http://www.microsoft.com/windows/netmeeting/, accessed November 2002.

Pixley, T. (2000). Document object model (dom) level 2 events specification. Technical report, W3C.

Qarbon (2002). Quick start guide to viewletbuilder. Technical report, Qarbon.com Inc., San Jose, CA 95113 USA.

Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. (1998). Virtual network computing. *IEEE Internet Computing*, 2(1):33–38.

Richardson, T. and Wood, K. R. (1998). The RFB protocol. Technical Report 3.3, ORL Cambridge.

Scheifler, R. W. (1988). *The X-Window System Protocol*. M.I.T. Laboratory for Computer Science.

Symantec (2002). pcAnywhere. URL, http://www.symantec.com/pcanywhere/, accessed November 2002.

Weber, B. (2001). Camtasia white paper: Video screen capture tool for windows. Technical report, TechSmith.