

Coordinated Visualisation of Video and System Log Data

Alistair Morrison, Paul Tennent, Matthew Chalmers
Department of Computing Science, University of Glasgow
{morrisaj, pt, matthew}@dcs.gla.ac.uk

Abstract

In this paper we present Replayer, a distributed, cross platform toolkit for utilising multiple coordinated visualisations in the analysis and understanding of heterogeneous data. In particular we analyse the methods used to combine recorded media such as video with numerical visualisations such as histograms and time series graphs. We examine in some detail the architecture behind the system, and the techniques used to maintain synchronicity and coordination when interactively brushing components. We demonstrate how Replayer can be used to explore data sets using an array of available visualisations, can focus analysis of video data on the most salient periods and can provide context for every area of the recorded data.

1. Introduction

As mobile computing technologies become more profuse, there is an ever-greater interest in evaluation and analysis of such systems. Evaluating the use of large, multi-user, mobile systems can be a challenging task. Onwuegbuzie and Leech [1] comment on the need to blend quantitative and qualitative analysis techniques to provide a coherent view of any data. The intention of Replayer is to present all these heterogeneous forms of data simultaneously to a collaborating interdisciplinary set of analysts in a coherent, synchronised whole.

The multi-user, distributed nature of ubiquitous computing systems often renders them unsuitable for study with many traditional usability assessments. Usability studies for non-mobile applications often focus on observation techniques, making use of lab-based trials and video analysis to evaluate participants' experiences with a system and assess their ability to successfully complete common tasks. There are a number of reasons why such methods can prove less useful for studies of ubiquitous computing systems. Video equipment in a usability lab setting would often be precisely placed in order to capture user activity. Evaluation of mobile systems will often be performed in a more natural setting, with non-stationary participants. This makes it

difficult to capture user interactions with mobile devices, or information being presented on small displays.

In a particular evaluation, it might also be important to record the activity of a number of users simultaneously. This creates a problem of managing a large amount of video data, especially if activity is monitored over a long time. Although data can readily be recorded by deploying several video cameras, it would be impractical for an analyst to sit through hours or days of footage in order to uncover potentially interesting incidents.

A study based solely on data recorded in system logs would also be insufficient to adequately evaluate a system. Although every system event and user interaction might be logged, it is hard to make such detailed accounts of environmental or social factors that might influence a user's behaviour. For example, a pause in interaction might be explained by a participant becoming distracted from the task, or a usage error might be due to lack of concentration while a participant speaks to a passer-by. The capturing of such contextual information is one of the strengths of video-based analysis.

Replayer [2] is a system designed to support both log and video analysis through the synchronised presentation of video data and information visualisation style data exploration tools. A number of visualisation tools are provided for the visual exploration of log data, allowing an analyst to summarise all statistical data from a trial, or focus on a particular factor of interest. The video data are synchronised with the log visualisations, allowing analysts to make selections in one view and immediately jump to the corresponding section of the video. As well as supporting a richer appreciation of the recorded data, the provision of these multiple views allows an analyst to gain a fast overview of the recorded events and perform time-consuming video analysis on only the most salient areas. Figure 1 shows an example of Replayer, displaying synchronisation between video and map components.

This paper focuses on the use of coordinated views in Replayer. Through a description of the system architecture, it is explained how audio, video and information visualisation brushing between separate components can be linked to support synchronisation and

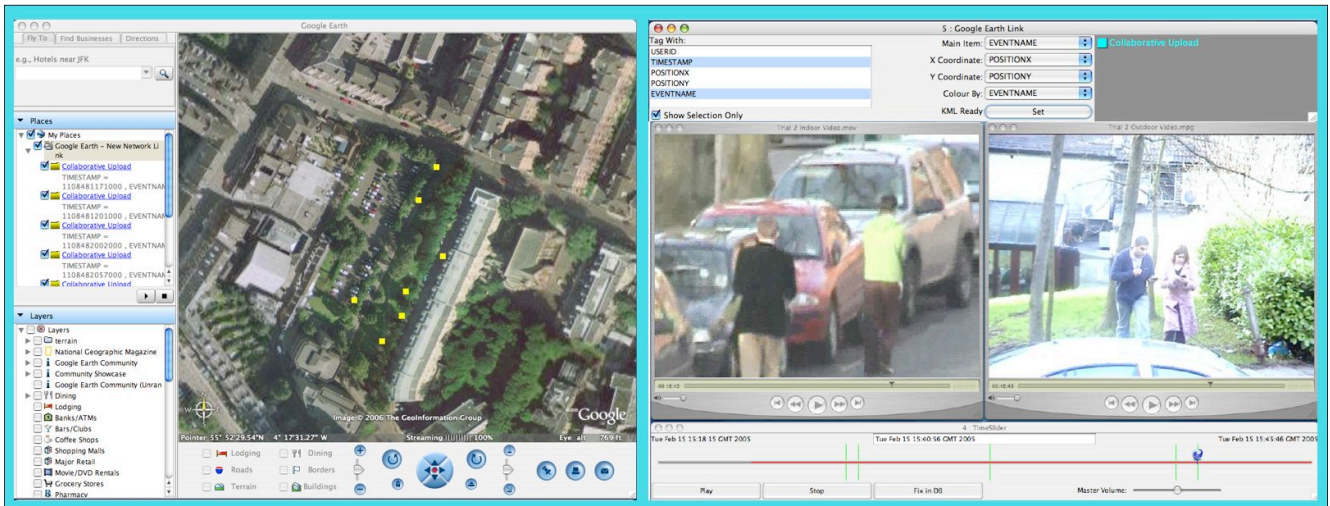


Figure 1. The Replayer system supports analysis of recorded activity from studies of mobile applications. The figure shows two of the visualisations provided in the Replayer system. A map component on the left illustrates the geographical spread of locations at which participants performed a certain system event (the yellow dots). The video tool on the right (including footage from two separate cameras) is synchronised with the map tool. The time slider control below the videos has several green lines corresponding to temporal distribution of the same system event. Playing the videos automatically jumps between the marked time periods, showing every occurrence of the event and highlighting the location on the map.

views and across different machines. A discussion is presented as to how such a coordinated environment allows sociologists and computer scientists to interact around a coherent visualisation that couples resources usually associated with just one of these two communities of research practice.

The remainder of the paper is divided into a number of sections. The following section summarises related work in this area. In Section 3, the Replayer system architecture is described, and then Section 4 explains the format of logs required for Replayer compatibility and introduces a tool designed to automate this process. Section 5 details the individual visualisation components, before Section 6 discusses synchronisation in Replayer. Section 7 provides a discussion of the topics raised, and this is followed by conclusions.

2. Related Work

There has been a wealth of applications offering multiple views of data. One set of input data can be processed in a number of different ways and give rise to numerous graphical configurations. As each view shows distinct features, their use in combination reduces the risk of misinterpreting the data's structure [3]. Recent examples of such systems include a coordinated display of different aspects of recorded data on gene activity [4] and the City 'O' Scope system, which provided multiple views of statistical information to compare world cities [5].

The coordination of multiple views allows still greater insight to be made into data. This is often achieved through brushing and linking [6], whereby a user's selection in one view will highlight the

corresponding subset of objects in the others. For example, a particular group of data might have been clustered together when processed by one component. Being able to select this region and immediately see how the same group has been handled by complementary visualisation tools can greatly increase an analyst's understanding of the inherent structure of the data.

One system which focuses on collaborative video analysis is Fraser et al's VidGrid [7]. This is a cross-site system where videos can be studied simultaneously and dynamically annotated by a number of researchers. The GRUMPS system from McLeod et al. [8] was developed to dynamically and invisibly record log data from any java program, into a common format by directly instrumenting the compiled bytecode.

Creating replays of systems is not unique to Replayer. One example of a system where a replay was created from the data the Savannah system – an educational mobile game that required children to act out the life of a lion in the Savannah [9]. A replay tool was used in the analysis, showing player location on a map and messages being sent between devices. The George Square project, a collaborative tourist support system, underwent a similar-style analysis. A map tool was displayed, along with photos taken at various marked positions, and location-based recommendations made to other tourists.

In these examples, however, new replay tools were built specifically for each system. This obviously required considerable time and effort in each case, and the created systems were tightly coupled to the evaluated system, offering limited potential for re-use. Replayer improves upon this by providing a generic extensible tool for such analyses.

3. System Architecture

This section details the Replayer system architecture. An overview is provided, then the roles of certain critical controls are explained, before issues particular to visualisation, networking and database systems are discussed.

3.1 Component Architecture

Replayer uses an extensible distributed component architecture. Each component of the system runs in its own process, communicating via a TCP interface with the server and is literally a separate program. Replayer uses a client-server paradigm to provide data from a centrally managed database to each of several visualisation components.

This system has a number of benefits: firstly distribution – any computer on the same subnet as the server is able to run visualisation components, with the same inter-component communication as those running on a single machine. A second benefit is extensibility – it is possible to add new components to the system at runtime. Stability is another advantage of this architecture – any programmatical error that may occur in any of the components cannot affect the others; it is therefore unlikely that current work will be lost if a third party component fails. A final benefit is cross-platform and cross-language support – it is possible to run visualisation components on different machines running different operating systems and still maintain the same level of interactivity. Indeed in some cases, components are written specifically to take advantage of the facilities available on a particular platform. Because the interface between components is that of generic TCP, components can also be written in any language that supports socket programming. The majority of the system is written in Java with Swing for cross platform portability, but certain areas are written in C# and Applescript.

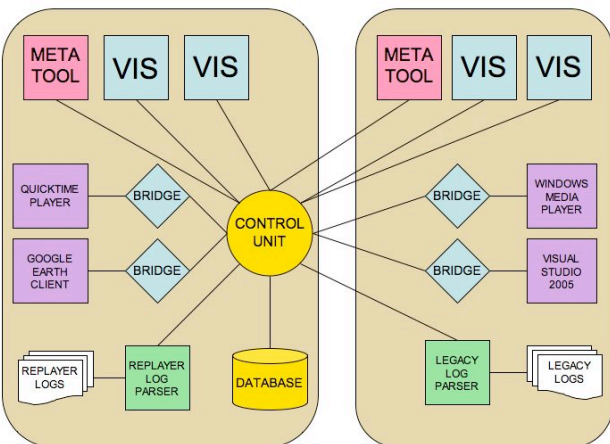


Figure 2. The Replayer system architecture. The shaded background represents two separate machines. A control unit runs on one machine and keeps all visual modules synchronised.

Replayer's central repository of data is an SQL capable database. SQL offers the user a very powerful way of querying the database to retrieve exactly the correct data for display in one of the visualisation components. Indeed, in some cases SQL queries alone are sufficient to retrieve much information about the data, such as counts of events or averages of numerical data.

3.2 The Control Unit and the Meta-Tool

The control unit acts as the central server for all Replayer's components. At its heart is a database management system built using the Hypersonic SQL system (<http://www.hsqldb.org>). This allows the database to be run in main memory, making it fast and, more importantly, offering an SQL-capable database that does not rely on another application, but can run within the control unit process. The architecture of the database will be discussed in Section 3.5.

Users do not interact directly with the server, but register commands through the meta-tool, as seen in Figure 3. This is due to the distributed nature of the system; only one server can be running at a given time for a particular set of components, so in order for users on all active machines to have the same facilities available, each is able to run a meta-tool. The meta-tools are kept synchronised automatically, so that any interactions performed in one are propagated throughout the network.

The meta tool provides various facilities for managing the Replayer system. A table at the top maintains a list of active components, along with the machine on which they are running and the queries used to load each with data. SQL queries can be composed in the text box at the bottom of the component and a column of buttons at the right provide a number of functions.

New components can be opened and seeded with data, and replacement data can be pushed at each component if required. The meta tool also allows users to examine the raw data tables. This can be of particular use when constructing appropriate SQL queries for components.

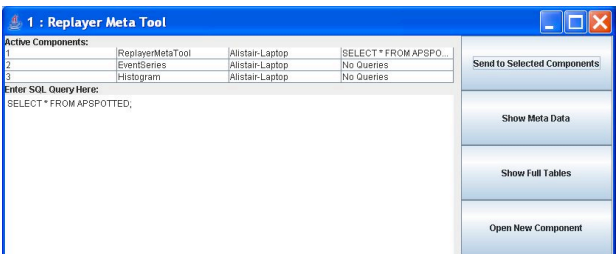


Figure 3. The meta tool component shows the open components and provides controls for viewing the tables or opening new components.

Clicking the Open New Component button displays a selection of tools. This list of the available tools is automatically generated by scanning for executable JAR (Java Archive) files. These files will contain executable code, classpath dependencies and a screenshot to add to the tool selection menu. Components can be written in any language, with the executable being placed in the JAR wrapper.

Using this scheme, Replayer is extensible at run time, as new components can be written and are immediately for use without the need for termination of a running system or re-compilation.

When a new visualisation component is opened it must be seeded with a subset of data to display. This can be achieved by directly typing the SQL query into the meta tool and sending that query to the new component. Alternatively there is the option of creating a visual query with the component loader, as seen in Figure 4.

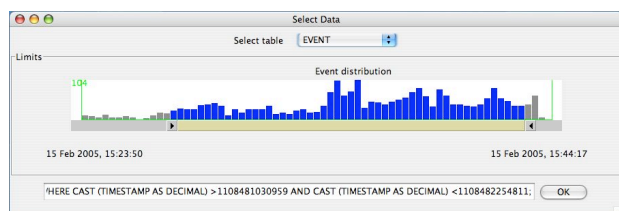


Figure 4. The component loader can automate SQL statement construction. Time-based data distribution is illustrated with a histogram.

The component loader is only capable of visually constructing very simple queries; anything more complex still requires SQL. However the user is able to select which table they wish to use from a drop down list, and constrain the time using a double ended slider. For convenience a temporal distribution of the events in the table is displayed in a histogram above the slider. The generated SQL is displayed below and can be edited if necessary.

3.3 Visualisation Components

Replayer contains several information visualisation-style components. These can operate in a stand-alone capacity to show certain characteristics of a data set, but show their full worth when operating in coordination via selection brushing. Each component will be discussed in more detail in Section 5.

Log parser components are also supplied. While Replayer and its supplied parser are intended to operate primarily on logs in the format specified and generated as part of the toolkit, it is often the case that users have logs in legacy formats. It is possible to build a component to parse these logs and record the data into the Replayer database – indeed using the same technique it is actually possible to update the Replayer database in real time from a currently running system.

There are a number of bridging components – used to connect a stream of Replayer data to the inputs of

other programs. One such component acts as a bridge between the Replayer data and Google's Google Earth client (<http://earth.google.com>). Another acts as a connection between the Replayer data and Apple's QuickTime.

While it is possible to construct a component which will examine or affect the data in any number of ways, some constraints are applied to the development of components. These are primarily concerned with ensuring the components meet the appropriate networking protocols. Replayer uses a simple grammar to distinguish between different forms of network data, and this is made available to component builders. Each component must register with the control unit when it starts up, inform the control unit when it shuts down, and be able to handle the input of data and selection information. Other than these four rules, component builders are free to implement the components in whatever manner they see fit.

3.4 Network Architecture

The majority of the communication in Replayer takes place over bi-directional TCP links between the components and the server. In some cases, such as that of the Google Earth component there is a further communicative step to connect to some third party program. A single control unit broadcasts its existence across a subnet on a particular port. In this broadcast are details of its IP address and the 'admin' port to which TCP requests for new connections should be made. A new component listens for that broadcast, and then makes a TCP connection to the control unit's admin port. It is then assigned a pair of TCP connections, one for sending and one for receiving – this is required because of the asynchronous nature of Replayer's communication.

While it may have been possible to achieve this effect with a single port, the intention was to keep the communication protocol as simple as possible, theoretically yielding the broadest range of additional components.

Once connected, a component will receive data and selection information when the control unit pushes data at it through one TCP connection, and may use the other to either request specific data or to send selection information. The control unit sends out administrative data to all components whenever a component starts up or shuts down, or a query is made. Most components ignore this administrative data, but some such as the meta-tool, may wish to maintain an awareness of its peers.

3.5 Database Architecture

The heart of the Replayer system is the database. The basic structure of the database remains fairly constant, irrelevant of the particular data populating it, though it may differ considerably in detail.

To properly understand the layout of the database, it is of benefit to summarise the types of data it will hold. These can be broken into four main areas: state data, event data, media and provenance data.

State data is defined by its continuity. It is a representation of the current state of a system – a regularly logged set of values, all of the same type. This can be represented in a single table.

Event data, conversely, refers to irregular data – it is a representation of a system event, or something that happens, rather than something that is. Because it is non-continuous, and there may be any number of different event types, this is stored differently. A central event table is maintained, containing timestamps, usernames, and the names of the event. Each event name corresponds to a separate table (assuming one is required) which stores the attributes associated with that event. An example of this type of data might be an event where the system being evaluated has recorded the detecting of a wireless access point. The main event table is updated with the username, the timestamp and the string “AccessPointSpotted”. Following this, the table called “AccessPointSpotted” is updated with the MAC address, SSID, and security type of the access point.

The next data type is media, which is a table containing references to all the recorded media from the system. This may consist of such data as videos, audio recordings of participants or audio field notes. Each media file is tagged with a timestamp, allowing it to be synchronised with the data, and a username, which connects it to the user it concerns.

The final data type is provenance. This is data that has no timestamp associated with it. Typically, provenance data contains information gathered from participants during the trial, and any required information about each participant. In a study of a team-based game, for example, it might contain information mapping each participant to a team.

All the data in the Replayer database is tagged with a unique Replayer Identifier (RID). RIDs are used to identify data quickly within the system for selection purposes – selections are passed between components as a list of RIDs to achieve the interactive brushing effect. RIDs are generated by the control unit, which modifies every incoming SQL CREATE or INSERT statement to include an RID. The reason this is handled by the control unit rather than log parser is one of management – it is possible to have data input from several parsers with no awareness of their peers, and therefore it would be challenging to decide which RIDs to use as primary keys for their data. The only case where duplication of RIDs takes place is in the events – where they serve as a foreign key to the main event table. This means that a component showing all the recorded events can nevertheless brush and affect one which shows only events from a particular database table.

3.6 Third Party Software and Exports

Replayer makes use of a number of third party software packages in its component implementation. There are two main reasons for this. First, it is the intention of Replayer to allow users to work with the tools they are experienced with as much as possible – the arguments for doing this have been much explored in studies such as those by Suchman [11] and Dourish [12]. On a more practical note, because of the breadth of techniques Replayer utilises, it would be time consuming to recreate systems which have been developed and refined over time into stable and versatile single task oriented systems. For example in the case of our video component, rather than create a whole new tool for viewing, editing, and exporting video and audio data, we simply leverage the power of the two most popular systems for their respective platforms: QuickTime and Windows Media Player. Equally, Google Earth provides us with a free and customisable map package which allows us access to extremely detailed maps of the entire planet – meaning any geographical data we may have can be displayed easily. It is also an intention to make as much of the data as possible exportable. All the visualisations will be able to save screenshots, and all the database data can be exported as comma separated value (CSV) files, which can then be imported into other software packages. Indeed the database itself is exportable if the user so wishes.

4. Formatting Input Data

One major area of developing a system such as Replayer is that of data. It must be collected, collated, parsed and written correctly into the database before it becomes possible to begin analysing it. Because Replayer encourages the use of logged data in a particular XML based format, Replayer offers programmers a tool called the Instrumentor.

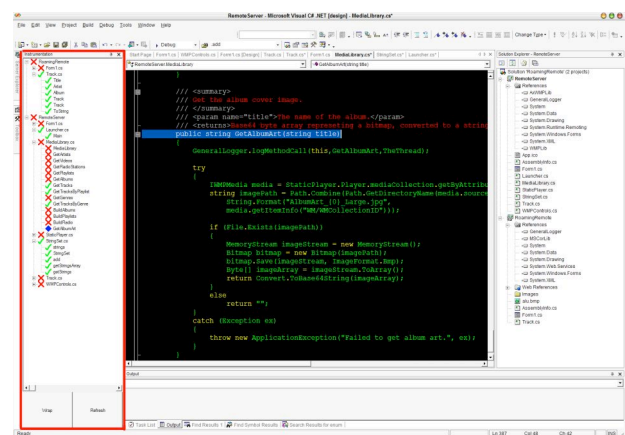


Figure 5. The Instrumentor Add-In to Microsoft Visual Studio (highlighted in red on the left).

The instrumentor, shown in Figure 5, is an Add-In to Microsoft Visual Studio, which inserts logging code

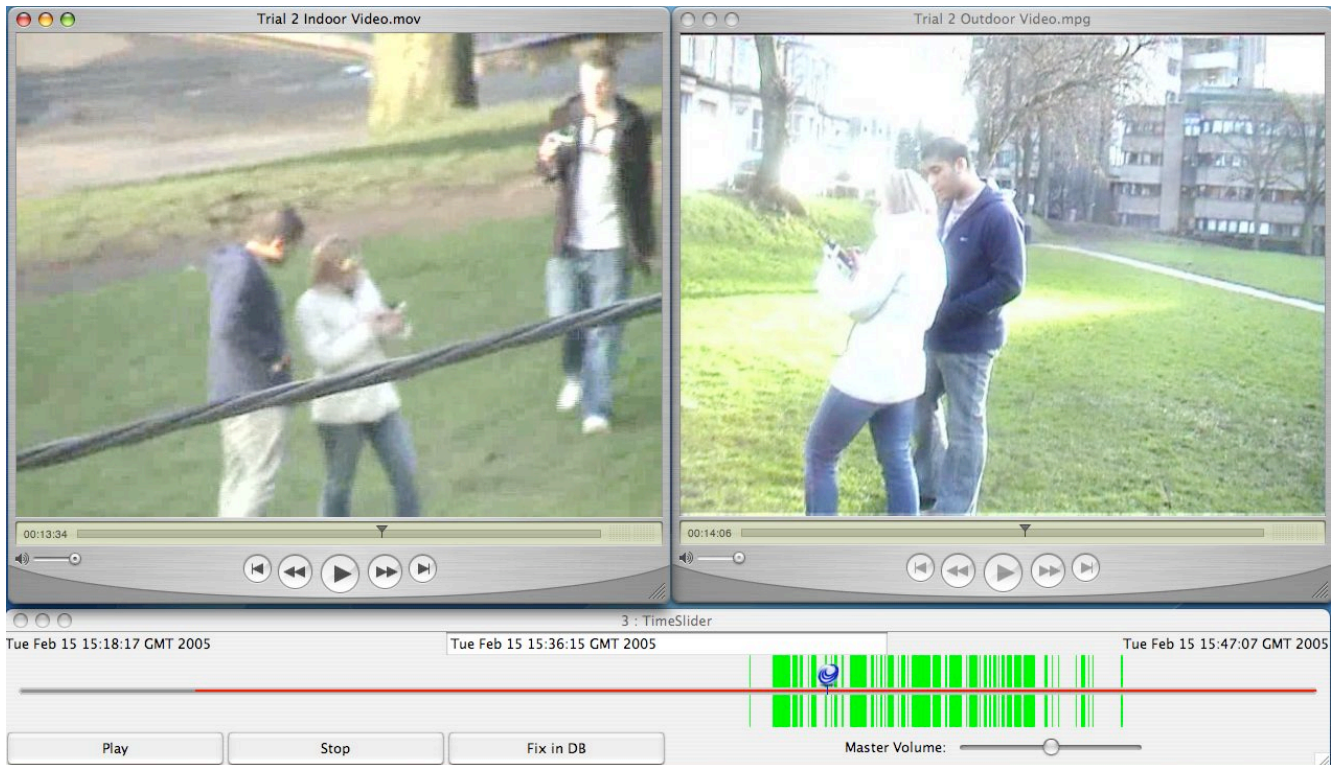


Figure 6. Replayer's time slider component is shown, along with two videos associated with a particular dataset. The area of the slider's track coloured red shows that area where media clips are available. The green blocks are those areas of time highlighted by making a selection in another tool. The thumb of the slider points to the current point in time being displayed on the videos. In these particular clips, the left stream shows a clip recorded from a camera in a building overlooking this trial. Just in shot is one of the field evaluators, holding a video camera. It is the view from this camera that we see in the video on the right.

directly and visibly into programs. Any instrumented program is thus guaranteed to be compatible with Replayer. Logged events are named with the name of the method being logged, and can be tagged with a more descriptive comment if required. Creating state data takes a little more care. This equates to regularly sampling the values of a set of variables. The Instrumentor creates a method called `log()`, containing the code required to output the values of those selected variables, and it is up to the programmer to establish a place in the program from where to call that method.

Once the logs have been collected, Replayer's parser converts each event or state sample to an entry in the database, dynamically creating new tables as required for each previously unseen event type.

While Replayer is principally designed to operate on logged data in this format, it is often the case that legacy data needs to be analysed, and cases such as these necessitate the creation of a new parser. This is simply a matter of generating SQL `CREATE` and `INSERT` commands and sending these to the control unit.

5. Component Modules

This section briefly introduces the components in Replayer's suite of tools.

5.1 Media Bridges

A large part of the Replayer system is the inclusion of media such as video and audio along with more traditional numerical visualisations in the dataset. To this end the Media component is included. This description will focus on the Apple OSX specific QuickTime Bridge component, the effects (if not the technique) of which are duplicated for Windows users based on C#'s ability to control Windows Media Player. The initial visualisation presented to the user is that of a timeline slider (see Figure 6).

This can be constrained from the full extent of the database, as with all other components by means of a suitable SQL query. The timeline slider then requests the data from the Media table of the database, and colours on the timeline the places where media is available. This alone can be important – a system trial may last many days or even weeks, and video clips may be sparse, so being able to locate them on a timeline is a useful visualisation.

It is also from this component that new media clips are added to the database. To add a video it must first be synchronised to the data. Video synchronisation is among the more frustrating and often challenging areas of qualitative data analysis. We use a tool called QCCI

(Quickie) at the stage of evaluation to help achieve this. QCCI is a field evaluator's tool written for windows mobile, and usually run on a PDA. It includes a number of functions to support the evaluation recording process, but the three most significant ones are these: a remotely synchronised clock display (Figure 7), a GPS log of the location of the carrier, and the ability to record timestamped audio field notes. Each of these have a purpose in Replayer for synchronising media clips. To time-synchronise a video, the camera is pointed at QCCI's clock, which has been synchronised to that of the system being evaluated. This shows the system time and allows the offset to be calculated between that frame and the point at which the video begins.



Figure 7. The frame from within a video, where QCCI's remotely synchronised clock was captured. It becomes relatively easy to synchronise videos with data using this simple lo-fi technique.

This simple lo-fi synchronisation is generally sufficient for our purposes, and in fact does not really require QCCI – it can be achieved simply by filming any synchronised clock in the video. Synchronising audio is a little more difficult. If the audio has been recorded with QCCI then it is already timestamped. If however it has been recorded on another device, it must be synchronised with the data using QCCI's timestamped clapperboard noise. To actually achieve this synchronisation the user has merely to select the appropriate frame in the media file, enter the time displayed and click the button marked 'Fix in DB'. A dialogue box will then appear, asking for the name of the evaluator – this allows geographical synchronisation to be achieved by relating the positional logs of that evaluator to that media clip. The path to the media clip, along with its start and end times, and the name of the evaluator are then written into the media table. This spatio-temporal synchronisation allows areas of the video to be selected in a number of ways. The timeline shows the selection by highlighting the relevant slices of time in green, as illustrated in Figures 1 and 6. When a selection is made from here, or from another component, the media clips will automatically jump to the first selected frame.

The display and playing of the clips, unlike the time slider component itself is operating system specific. On Apple's OSX, QuickTime is used, and controlled from the timeline slider using the Applescript application control protocol. This allows Replayer to directly control any action available within QuickTime. We are thus able to leverage this extremely powerful media tool to display and playback many media clips simultaneously, and to show and hide media windows as necessitated by their relative positions within the data. All are initially opened and hidden as their playback is time critical.

In initial discussions with evaluators and analysts, one of the requirements highlighted was the ability to 'scrub' through the data, that is move the thumb along the timeline and see the selection (and media) change as this happens. However, in most cases, it is not desirable for moving the thumb to actually make a selection in the data, but rather to simply move the frame in the media clips. A separate 'scrub' mode was thus implemented to achieve this, and has proved to be a powerful feature.

Making the correct choice of video and audio codecs is extremely important when saving media clips for use with Replayer. Playback of video is a CPU intensive process, especially in the cases where high compression is used. Playing several of these high-compression videos simultaneously will result in slowdown, which will in turn throw the synchronisation off. However, because of the distributed nature of Replayer, it is possible to be playing different videos on different computers, thus minimising this effect and allowing the user to progress as normal.

5.2 Time Series

A tool has been incorporated into Replayer to plot various time series data (Figure 8). Each numerical data variable provided as input is drawn as a line on the plot, with colours related to variable names by a key on the right. Axes are labelled automatically, and re-labelled if the time series module window is resized.

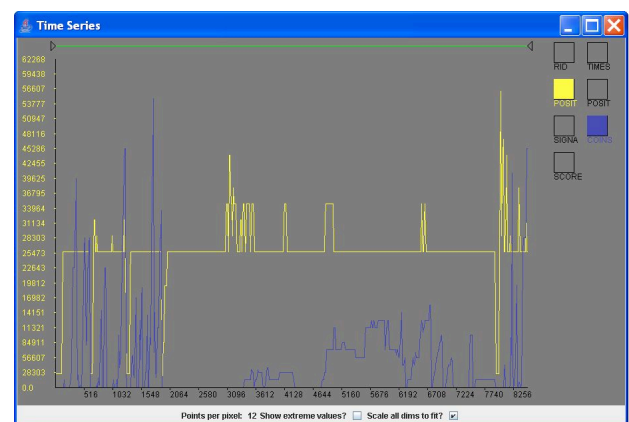


Figure 8. The time series tool graphs a series of data streams by time. The tool can compare attributes between users or view a summary of a single users' performance.

The time series module can be used in combination with other visualisations, to allow selection and linking as described in Section 2. A double-ended slider is provided at the top of the module, with which a specific section of the series can be selected. If the data are also being viewed in another connected component, the system will highlight the corresponding time measurements. Similarly, marking time periods in other views will alter the time series display to reflect the selection. In Figure 9, the time series is reflecting selections made in another component. The IDs of the objects contained in the selection have been passed to the time series so that the corresponding areas of the could be highlighted. The time series variables are greyed out at the deselected periods, with the background also darkened.

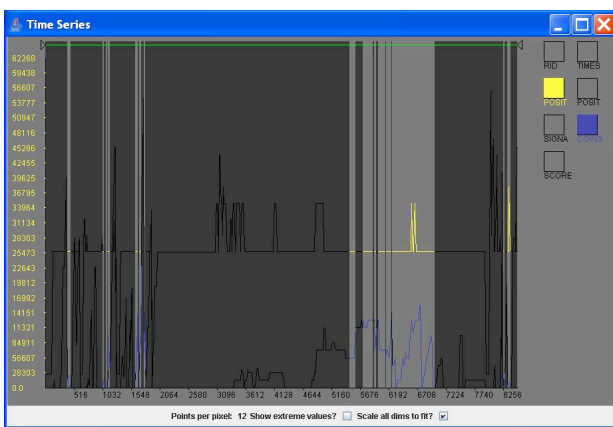


Figure 9. The time series has received a selection and has shaded the de-selected periods.

5.3 Event Series

The time series module shows system state over time. While this is a useful tool, it is unsuitable for much of the data that is likely to be recorded during a system evaluation. State data is continuous and represents properties that will have a specific value at any instant of a system trial. It would also be of benefit to study event data, which describes discrete events that happen periodically throughout system use. Examples would be user interactions such as button clicks, or server messages.

The event series has the same time-based x -axis as the time series, but visualises these discrete events as icons. Figure 10 shows an example, displaying data recorded over a week-long trial. In this case, each event is the discovery by a participant's PDA of a new wireless access point. Certain pieces of information are logged each time this occurs, which the user can explore with controls provided on the event series, as described below.

The tool is useful in providing an immediate overview of a whole trial, while allowing users to zoom to particular periods of interest. A context window in the

top right of the tool shows all of the data currently loaded into the tool, with a green window illustrating where the current focus fits into the overall context. The green widow can be dragged around in the context view to immediately jump to another area of interest. The view in the figure is zoomed to show events taking place over two days, but there is a degree of overlap that could be resolved by zooming in to an hour or a few minutes of recorded data.

A drop-down list is provided at the bottom-left of the tool to allow users to select the input dimension with which to plot the data in the y -axis. If a numerical dimension is selected, the y -axis will be scaled appropriately and each event will be drawn at the appropriate height in the frame. On the other hand, if a nominal dimension is selected, the number of unique nominal values on that dimension is calculated, these are spaced out evenly on the y -axis and objects are placed accordingly. The figure shows a selection of "SSID": the owner-assigned name of the wireless network. Moving the cursor over any object in the event series generates a tool tip showing the exact value on the x and y dimensions.

At the bottom-right of the tool is another drop-down list, to determine the input dimension with which to colour objects. In the figure, the user has selected "NAME", corresponding to the ID of the participant whose PDA generated each event. A key is provided, as in the time series, which can be used to filter data in the same manner.

Selections can be made in the event series by dragging a box around the objects of choice. Non-selected items are then greyed out.

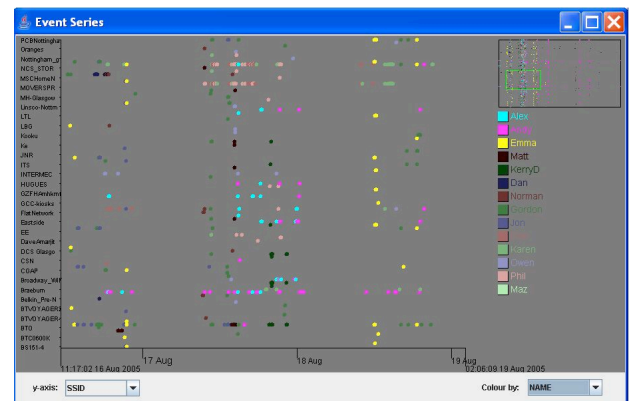


Figure 10. The event series tool plots discrete events by time. In this example, the y -axis has been set to display owner-assigned names of wireless network access points, and each event is coloured by user name.

5.4 Histogram

The previously described components afforded temporal-based distributions, and judged events and states on properties at a given time. Replayer also

contains a histogram tool, which provides a means of assessing data by distribution. Rather than showing individual events, a summary is given of the aggregated measurements. Figure 11 illustrates.

Like the event series, the histogram tool has a drop-down list from which an input dimension can be selected. The histogram will then display the distribution of values recorded on that dimension. Should the selected attribute be based on nominal data, a separate bar in the histogram is created for each unique value. This is the case in the figure, which illustrates the frequency of each distinct event. If the selected dimension contains numerical data, the data will be bucketed by value. Tool tips are used to present the x-axis labels, with the appropriate nominal being displayed when the cursor is moved over a bar. This action also shades the bar red and highlights its height in red on the y-axis, for easy value comparisons.

As with the other components, support is provided for brushing between views. Selections can be made by clicking on individual bars, which will instruct the histogram to send other components the IDs of the objects represented therein. The histogram visualises received selections by shading certain amounts of each bar. Colour is filled in proportion to the amount of objects represented in the bar that are selected. In the figure, every object represented by right-most bars is included in the selection, whereas less than half the values in the bars to the left are selected.

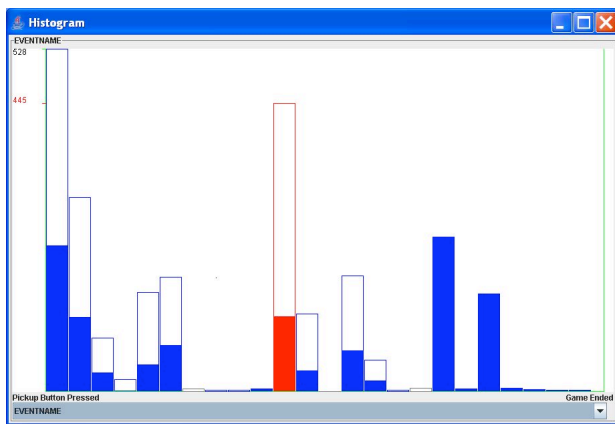


Figure 11. The histogram shows a distribution over time. This example shows the occurrence of each of a number of events. Bars are filled in proportion with the number of each such event that exists in the selection.

5.5 Google Earth Bridge

Replayer's Google Earth Bridge component works by creating keyhole markup language (KML) files, then serving them to Google Earth over HTTP. Google Earth provides two main techniques for retrieving these files – the refresh can be manual or automated.

The automated refresh means that the data is periodically updated, making this ideal for creating

animations, while the manual refresh is better for making selections. When a refresh is requested, as part of the HTTP GET request Google Earth sends the current 'view box', that is, the geographical points at the corners of the currently visible area of the map. By examining these points, the Google Earth Bridge component is able to establish the RID of each datum being viewed and send these out as a selection to the other components.

From a display point of view, there are two different ways data can be shown: statically or animatedly. Static displays are used to show spatial distributions of events, while animation is ideal for showing simultaneously with videos – showing the location of users moving around on the map. The system is able to show data of more than three dimensions (latitude, longitude and altitude) by first colouring the points on the map by one nominal, for example by user name. Additional dimensions of data can be included by making use of the description bubbles provided for each marker on the map though these are only visible one at a time.

Figure 12 shows the tool.

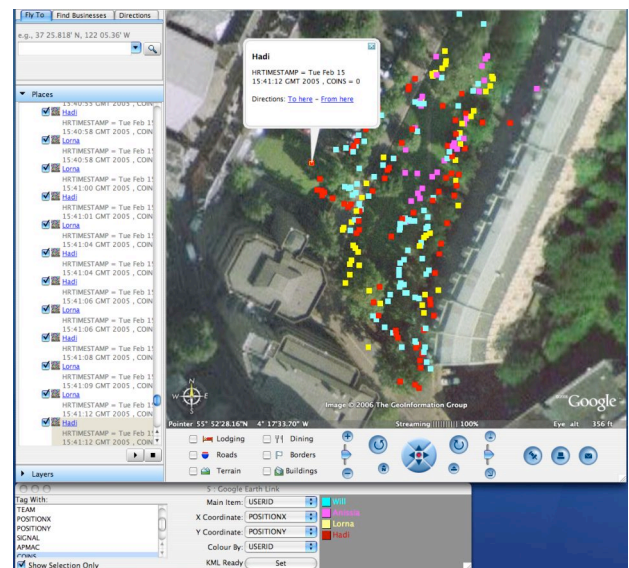


Figure 12. The lower part of this figure shows the Google Earth Bridge, which shows what data is to be included in the KML, and a key to the marker colours. Above is Google Earth showing a spatial distribution of participants' positions for a particular criterion. On the left of the Google Earth application is a list view of all the markers, including additional dimensions of information. Any point can be selected and focused on in either of these views.

5.6 Example Of Use

Figure 13 shows Replayer in use, running on two machines, each with additional monitors attached. This example is analysing data recorded from Treasure – a multi-player mobile game that exploited seams in wireless network connectivity [13]. Each player was

given a PDA showing a map, upon which was scattered a number of coins. Players had to walk to the physical locations at which the coins were represented to ‘collect’ them, and then move within wifi connectivity to register their haul to the server and score points. Players could steal coins from opponents before they uploaded, but only while both parties were in network the range. In exploiting patchy network connectivity, the project sought to make a positive use of what is generally considered to be a negative phenomenon.



Figure 13. A photograph of Replayer running on two machines, each with an additional monitor. The two computers are running different operating systems. Five tools are in operation, and all are operating in coordination. Figure 14 provides further detail.

Figure 14 shows the various components from Figure 13 in greater detail. Starting at the top right, an event series is displaying wireless network signal strength over time for each of four users. The data has been sampled every second over a roughly 25 minute period. It can be seen that there are a number of points forming a horizontal line significantly lower than the main body of data. These represent the periods at which there was no wireless signal. A rectangle has been drawn around the upper objects, resulting in all the data captured with a network signal strength of zero being shaded grey and deselected.

This selection is reflected in the other tools. The map in the bottom left shows the game area around the Computing Science Department of the University of Glasgow and has plotted player positions based on GPS (Global Positioning System). The wireless access point that the game used, housed inside an office in the building, is shown on the map as a dark blue marker. Following the event series selection, this visualisation now also shows only those events where the participants had a network signal. The map therefore reveals the curved perimeter of signal availability¹. Although it is not shown in the figure, selecting the opposite set of objects in the event series would show roughly the same

perimeter, but with the rendered positions being those recorded while outwith wireless range.

Moving on to the video component in the top left, it can be seen that there are two streams of footage. The left view was recorded from a fixed position at a window in the building, while the second stream was recorded by a field evaluator who walked around the game area amongst the participants. The time slider directly beneath the videos has been highlighted in accordance with the selection, with the green areas representing the selected periods of time. The analyst has moved the blue control thumb to a non-selected area towards the end of the trial. The videos then jump to this location, and it can be seen that the game participants were in a car park at this period. This car park can be seen in the top centre of the map, just outwith the network perimeter.

Finally, turning attention to the visualisation components in the bottom right of the figure, some thought can be given to player tactics during the game. The time series component is plotting the number of coins held by one particular user over time. It can be seen that this value gradually builds up, as coins are accumulated, and then falls instantly to zero as the coins are either banked to the server or stolen. Examining the shading from the selection, it can be seen that during times of connectivity coins were collected in very small numbers, relatively infrequently, and uploaded instantly. In contrast, the shaded areas show times when the player had a large numbers of coins. It seems that the player would not generally leave the networked zone to collect just the one coin, but would be drawn out for larger rewards. It can also be seen that the coin tally reduces to zero immediately on each occasion that the player re-enters connectivity.

The histogram confirms this analysis of user behaviour. It charts the distribution of coins held over time, taking data from all four participants. The state of having zero coins was by far the most popular, but was filtered out using the SQL query and does not appear in the figure. Unsurprisingly, this leaves one coin held as the most common state. Looking at the effect of the selection on this visualisation, it can be seen that the majority of cases where a user held one or two coins occurred during network connectivity, corroborating the findings of the time series, where users would collect small numbers of coins and upload while in range, without building up large collections. As the number of coins increases, however, the proportion of the bar belonging to the selection decreases. Holding a high number of coins in the networked area is a risky strategy, as the player is left open to theft; it can be seen that the players realised this, and of the sampled occasions when a lot of coins were in a player's possession, far more time was spend out of connectivity than in.

¹ A few outlying objects can be explained by GPS error.

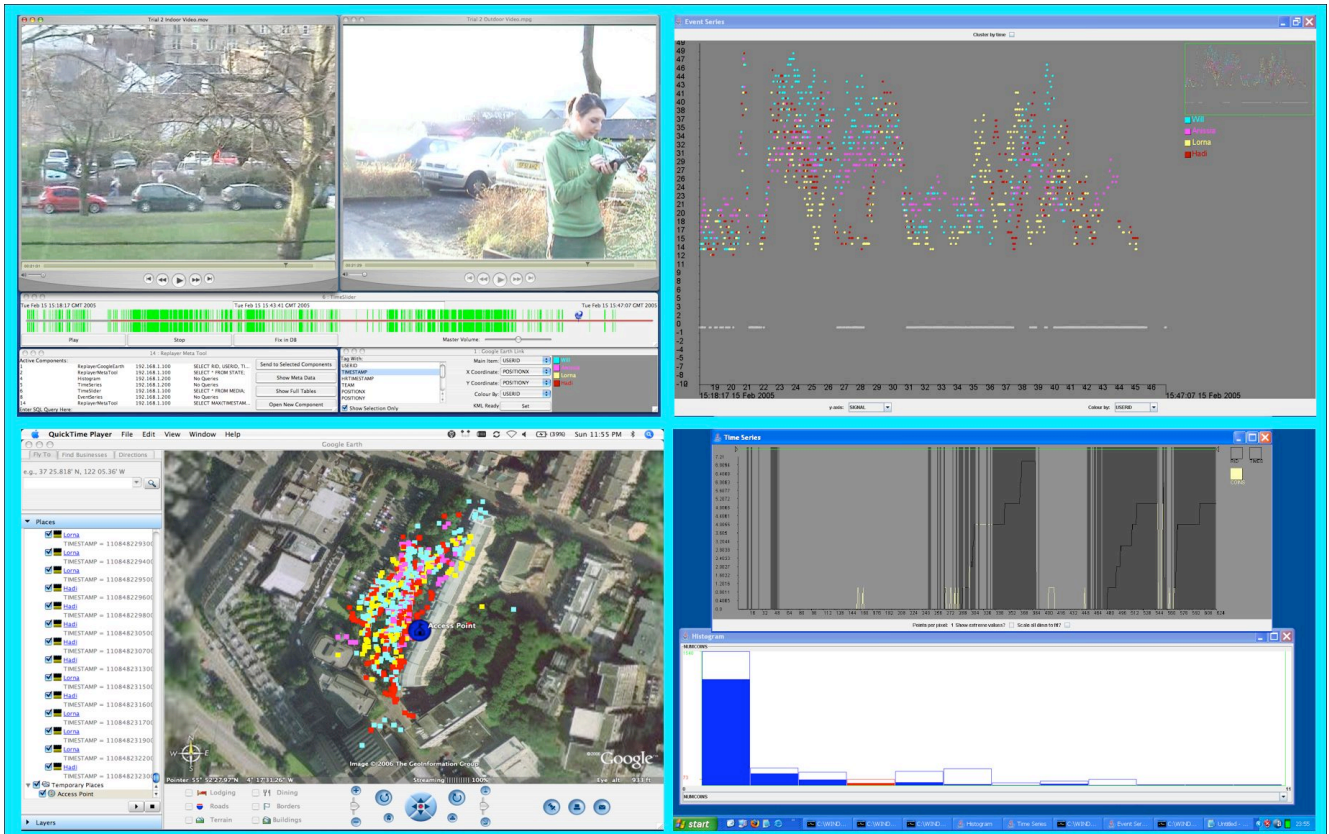


Figure 14. Five Replayer tools operating in coordination. Clockwise from top left, the figure shows the video component handling two streams, the event series charting signal strength for each user over time, a histogram and time series showing summary information on a system property and the map showing the recorded positions of users based on GPS. Data is taken from a multi-user mobile application.

6. Synchronisation

With many different visualisations of heterogeneous data types, the challenge of keeping everything synchronised is considerable. Replayer is a distributed network system, so all inter component communication must take place over TCP network connections. Additionally, because components are non-language-specific any communication must also be based on a simple grammar. There are two specific areas of the system where synchronisation must be maintained. The meta tools must remain synchronised showing both the current list of open components, and the queries currently associated with them. Secondly, the current selection of data between components must remain accurate, including when new components are opened.

Synchronisation of the meta tools is maintained with messages from the control unit. Whenever a new component, including a new meta tool, is opened or a query is sent to the database, the control unit broadcasts a list of currently active components. Included in this list are a unique identifier for each component; the type of each; the location of each, that is, what computer it is running on; and the last query made by each. The meta tools retrieve this list, and update their display

appropriately.

Synchronisation of selections is also maintained by the control unit. Data is referred to in Replayer primarily by its RID. Each entry in the database has an associated RID. When a selection is made in a component, it sends either a list of selected RIDs, or start and end timestamps, depending on the nature of the component making the selection. The control unit then calculates the converse, that is, if the control unit receives a list of RIDs it calculates the start and end timestamps, whereas if it receives timestamps, it calculates the appropriate RIDs. The control unit then broadcasts this pair of data to all the components. Again depending on the nature of each component, it will make and display its selection based either on the list of RIDs or the timestamps.

7. Discussion

Replayer is a tool which attempts to bind together lots of heterogeneous data. It encourages different styles of evaluation to be used collaboratively, aiming to bring together the best practices of both highly quantitative data-oriented numerical analysts, and more qualitative data-oriented researchers such as ethnographers. Taking these different approaches to analysis separately, both have strong merits, however when brought together, each

can support the other to create a style of analysis greater than the sum of its parts.

With the intention to encourage collaboration, comes a necessity to understand the different ways an incident is approached by different types of researchers. There are two distinct approaches supported by Replayer. The first is question-led analysis, where the researcher has a particular question in mind he or she wishes to have answered by the data. Replayer supports this technique by providing a wealth of different views for examining data, allowing almost any question to be answered, assuming the answer lies within the recorded data. The converse of question-led analysis is exploration-led analysis. In this case, the researcher does not really know what he or she is looking for, and will simply look for interesting incidents in the data; this is particularly relevant for video analysis, where the researcher may watch all the way through a video looking for interesting areas. It is also often applied to interviews and questionnaires. Replayer supports this technique by providing additional views on the data, which can help to give context to videos, or in the case of non-synchronised data, such as interviews, Replayer can be used to support, explain or in some cases, directly contradict what is being said by participants.

Another effect of bring together a cross discipline group of analysts is the motivation behind making Replayer distributed and cross platform. The intention is to bridge to the kind of tools users are comfortable with. A Replayer user should be able to arrive into a group of other Replayer users, start up their own computer, of whatever type they have a preference for, join the local network, and instantly begin manipulating the data across the whole group of machines. Equally we aim to support the use of as many commonly used software tools as possible, with direct control in some cases, and more limited export-based support in others.

Conclusions

We have presented the toolkit Replayer: a powerful, distributed framework for understanding heterogeneous data. We have explored the underlying architecture of the system, some of the diverse range of components, and the techniques used to maintain synchronicity between them. We examined the process involved in creating system logs compatible with Replayer, and showed how legacy data can be imported if necessary. It is the aim of the Replayer project to encourage at worst communication, and at best collaboration between researchers from different backgrounds examining the same data. It is with the blend of facilities for quantitative analysis from logged data, and qualitative analysis from recorded media, that it becomes possible to surround any single datum with much context, aiding the understanding of the researcher and thus improving the evaluation of the system. The distributed, cross platform nature of the system makes it generally usable by any researcher with almost any computing equipment, and allow users to effectively leverage the particular

strengths of each platform. We have shown a number of examples of how by making use of multiple views of the data we are able to explain that which may not be immediately apparent from a single view.

References

- [1] A. Onwuegbuzie, N. Leech, On Becoming a Pragmatic Researcher: The Importance of Combining Quantitative and Qualitative Research Methodologies. *International Journal of Social Research Methodology*, 8:5, pp. 375 – 387, 2005.
- [2] P. Tennent, A. Morrison, M. Chalmers, Replayer: Bridging the Qualitative-Quantitative Divide, *Under Review*.
- [3] J.C. Roberts. On Encouraging Multiple Views for Visualization. *IEEE Conference Information Visualization IV '98*, 8—14, 1998.
- [4] P. Craig, J. Kennedy, A. Gumming, Coordinated Parallel Views for the Exploratory Analysis of Microarray Time-course Data, *Coordinated and Multiple Views in Exploratory Visualization*, 3 – 14, 2005
- [5] D Brodbeck and L Girardin. Design study: Using multiple coordinated views to analyze geo-referenced high-dimensional datasets. In *Proceedings of International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 104--111, London, UK, 2003. IEEE Computer Society
- [6] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29:127-142, 1987.
- [7] M. Fraser, G. Biegel, K. Best, J. Hindmarsh, C. Heath, C. Greenhalgh, and S. Reeves, Distributing Data Sessions: Supporting remote collaboration with video data. *Proceedings of the 1st International Conference on e-Social Science*, University of Manchester, 2005.
- [8] I. E. McLeod, H; Gray, P; and Mancy, R., "Instrumenting Bytecode for the Production of Usage Data," presented at CADUI, pp. 185-196, 2004.
- [9] Benford, Rowland, Flintham, Drozd, Hull, Reid, Morrison, Facer, Life on the edge: supporting collaboration in location-based experiences, , *CHI2005*, pp.721-730, 2005.
- [10] B. Brown, M. Chalmers, M. Bell, I. MacColl, M. Hall, I. MacColl, P. Rudman. Sharing the square: collaborative leisure in the city streets. *Proceedings of ECSC*, p427-429. Springer, 2005.
- [11] L. Suchman, *Plans and Situated Actions*: Cambridge University Press, 1987.
- [12] P. Dourish, *Where the Action Is: The Foundations of Embodied Interaction*: Cambridge: MIT Press, 2001.
- [13] L. Barkhuus, M. Chalmers, P. Tennent, M. Hall, M. Bell, and B. Brown. Picking Pockets on the Lawn: The Development of Tactics and Strategies in a Mobile Game. *Proceedings of UbiComp 2005*, Tokyo, Japan, p358-374, 2005.