Software Replay Tools for Time-based Social Science Data

French, A., Greenhalgh, C., Crabtree, A. , Wright, M., Brundell, P., Hampshire, A., and Rodden, T.

School of Computer Science & IT, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK.

Email address of corresponding author: apf@cs.nott.ac.uk

Abstract. This paper presents the motivation, development, example use and future plans for our software tool which permits the replay, synchronization and annotation of heterogeneous time-based social science data sets. This 'Replaytool' is accompanied by a data management and visualization tool to make the process of compiling and playing back data sets easier for the user. The design and development of this tool has taken place as part of the National Centre for e-Social Science's *DReSS* node at the University of Nottingham.

Replaytool allows the user to synchronize and playback log files, multiple videos, spatial data etc., using a flexible system of 'viewers' which can be extended for new data types. Free-text annotations can be added during playback. These are stored in a relational database or an accompanying text file.

Current development is focusing on migrating to a Resource Description Framework-based database to allow greater flexibility in the annotation system, the recording of media and project meta-data, and support for the development and application of coding schemes. The new system will use a client-server web service approach, with partial replication at the client(s), supporting both a central (workgroup) store of data and media files and also local copies for offline working.

Introduction

The ability of a researcher to comprehend their time-based data can be greatly aided by the features and usability of the tools available to replay, manage, annotate and visualize that data. Data is nothing without understanding, and having a good understanding means being able to access and manage that data in a way meaningful to a researcher. Work in the DReSS ('Understanding new forms of Digital Record for e-Social Science') NCeSS node is examining this area and developing a prototype tool to address these issues.

The ability of software tools to help the researcher in the analysis of social science data should not be underestimated. Consider the mechanical analogy of the VCR. The data on the

magnetic tape is nothing without a suitable tool to visualize it, and the VCR provides this functionality as well as additional usability enhancements that allow the operator to move through the tape to see the required sections of the data. This facility is now second nature to most, and operation of the VCR forms an important stage in analyzing the data without requiring much conscious effort from the operator, who can concentrate on examining the data itself. Usability is key here, as such tools become irritating or impossible to use if their functionality is non-intuitive (take setting some old-style VCRs to automatically record programs as a case in point here!). What this analogy shows is that, even before software tools, other hardware tools existed for viewing and managing data, and today they often have analogies in the software world. These new tools are able to offer extended functionality over the original tools, but the key is for them to remain easily usable. Such tools equip the social scientist with the capability to manage the datasets of ever increasing size and complexity generated by modern research.

Current methods

There are many existing software tools which are today in use by social scientists to aid the understanding of their data. These tools can be divided into two classes: those which perform a specific task and those which are general in nature. Examples of the first class of specific tools include those which support the annotation and visualization of particular kinds of data, such as text, audiovisual or still images – but typically not all three. This software does a specific job on a specific subset of data types.

Into the second class fall tools such as word processors or video editing software. These kinds of software tools are quite general in application and are often used as organizational or lab book-like tools. For example in ethnography, previous work has suggested word processors and spreadsheets can be used by the researcher to build a description of the events taking place (Crabtree, 2006), and in other domains video editing packages have been used to synchronize and replay multiple videos. However, sometimes these tools are pushed beyond their intended use: for example not all video editing packages allow the user to view multiple videos at once, and the user may have to decide which video they are looking at during any one time period. Additionally, although annotation of videos may be possible with these existing packages (Saferstein, 2004), often this functionality is not developed to the maturity required by most social science researchers. However, such use demonstrates an appreciation of the possibilities of new digital methodologies for replay and analysis. It is clear that as social science researchers are pushing the limits of the existing tools, there is a need for more appropriate, bespoke tools to fill the gap generated by these requirements. This is where software such as Replaytool is required. Combining the functionality of the specific tools with the flexibility of the general tools gives to the user a more holistic method of managing, replaying and annotating their data.

Replaytool implementation

This section describes the development and architecture of the Replaytool software for synchronized playback and annotation of heterogeneous time-based data. Replaytool provides users with the ability to synchronize heterogeneous time-based media sources (typically video and log files), play this back and navigate the playback using VCR-like controls, and add annotations at the current point in time. It is a tool whose development originated during the VidGrid project to aid existing video-based ethnographic methods of

data analysis, but has now become a general framework for replay and annotation of various media formats.

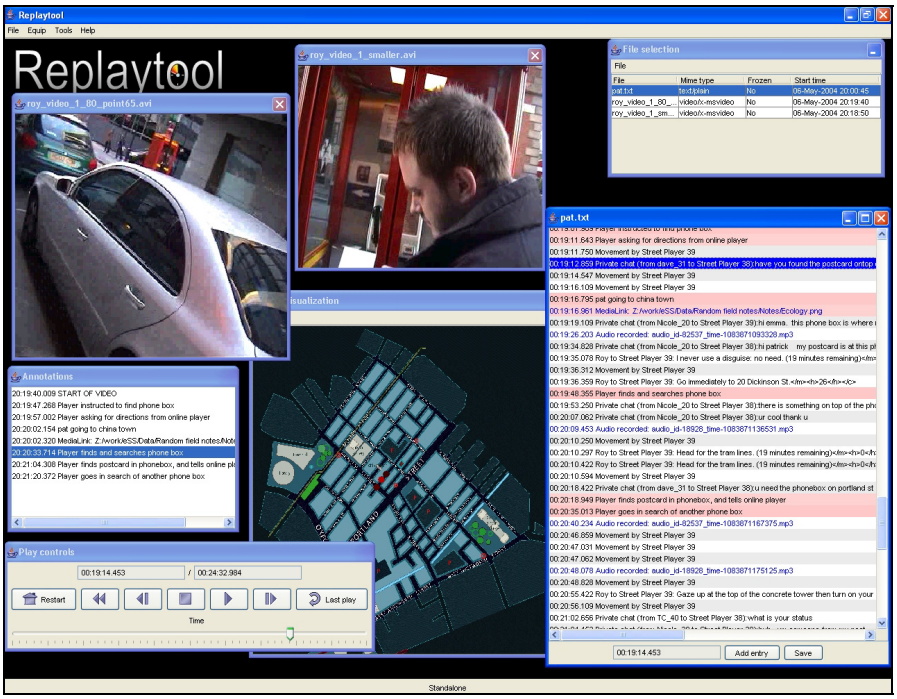


Figure 1. Example of Replaytool playing back a log file, two videos and location information, and the accompanying annotations.

Architecture

Replaytool has been designed for the replay of time-based data. The time dimension can be navigated using familiar VCR-like controls (see Figure 1), which allow steps forward and back in time, and replay in real time. There is also an option to replay the last section of time viewed. Data is replayed in a 'viewer window' appropriate to that data type. As time changes, the views in all the open viewers are updated accordingly. It is also possible to set the time from the viewers themselves; for example, clicking on a line of a system log file (such as that on the right of Figure 1) will shuttle the time to when that line was recorded. Another example is on the map viewer, where a user may click on a location and receive the times of all events logged as happening at that location – selecting one of these events again sets the replay time to that point. Internally there is a central time manager which maintains a clock, and all viewers are slaved to this clock (updating their views as it changes) while some can also set this clock (causing all viewers to jump to the chosen point in recorded time).

Of course with a set of media files, there is a need to make sure the files start playing at the correct times. For example, a computer may start logging events to a file at a particular time, but the researcher may not start a camcorder until ten minutes later. In Replaytool, files are synchronized by freezing one or more files, and moving the other files to the appropriate start times, thereby implicitly setting time offsets between the files. In the example just used with the system log file and the camcorder which is started after ten minutes, the video file would be frozen and the log file shuttled forward by ten minutes. The video would then be unfrozen. From this point on, the video file would be offset by ten minutes from the log file. There are a number of ways to make this process easier. Using the supplied management tools inside Replaytool (described later) the user can have some automated help in synchronizing files if

they know the real times when they were recorded (when the camcorder began recording, for example). The time control framework is illustrated below in Figure 2.

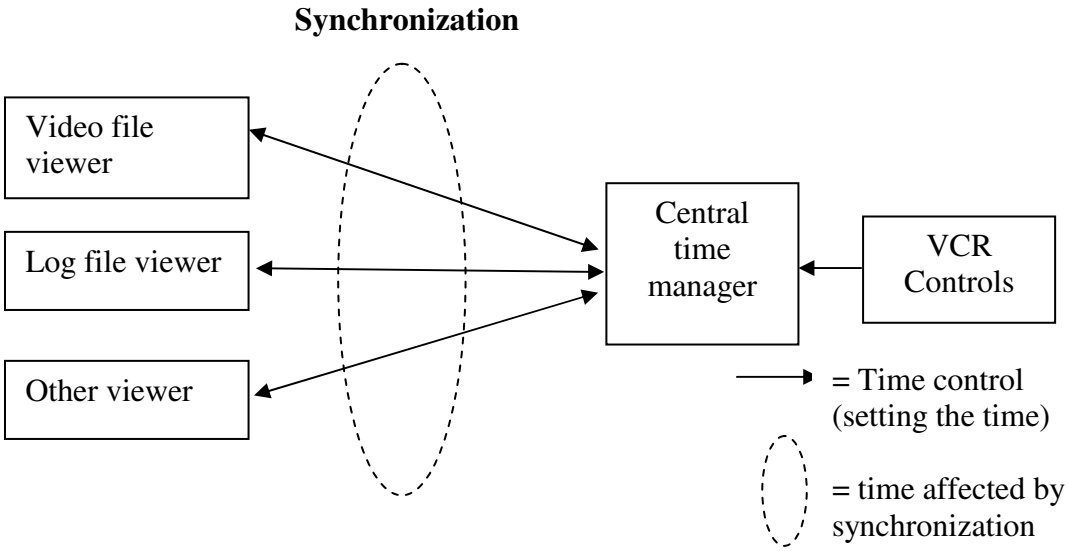


Figure 2. Time control flow for Replaytool

Currently, Replaytool allows the user to create annotations keyed to moments in time to coincide with the playback of the media set. Users can click a button to add a segment of free-text annotation or a hyperlink into an annotation file or to a database.

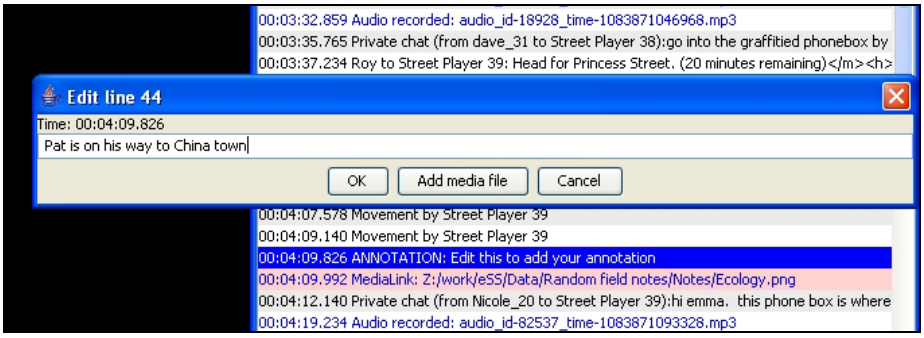


Figure 3. Adding an annotation.

The database annotations are currently stored in a MySQL database, the structure of which must be set up prior to Replaytool use. Annotations are pushed to and pulled from the database using a simple JDBC driver. Therefore, it is possible to keep the database on a separate machine, and access the annotations remotely. However, at this stage this is not recommended for multiple users at once as no database transaction management has been implemented.

Annotations stored in the relational database and are automatically inserted into log file data viewers – note that the annotations are inserted into the text 'on the fly' and are not part of the original log files – this means that the annotations can in theory be replayed against any time-series data source, interleaved at the correct times (as subtitles on video for example, or along an annotation track on a track-oriented view). Database annotations are also presented in a time-ordered list in a separate window:

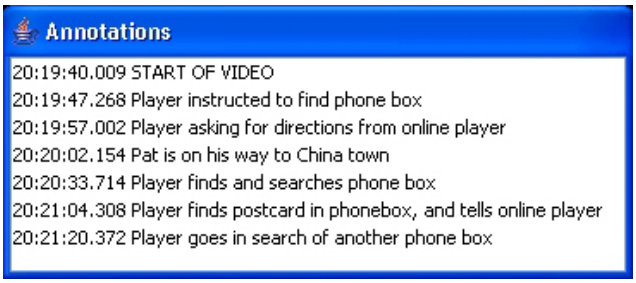


Figure 4. Automatically managed annotation list

This annotation index is managed automatically as users add annotations into other data sources. It allows an ethnographer to produce an index into the digital record itself, and provides an overview of extracted features of interactions.

Annotations which are hyperlinks allow a user to establish an explicit link to a related media file, rather than entering pure free-text. This linked media file can then be opened by double-clicking the annotation link in Replaytool. This allows external data such as maps, photos, notes etc. to be incorporated into the replay framework at appropriate points.

Current development effort is on expanding the versatility of the annotation system by permitting the annotation of many more types of data – no longer just points in time. A new ontology is in development to allow the description of Replaytool projects, and one section of this handles the annotation of data. This new annotation framework will allow the user to apply structured annotations as well as the existing free-text annotations.

The purpose of Replaytool is not to reinvent the wheel: other software is available which provides the social scientist with ways of statistically analyzing their data, capturing video, etc. Clearly there is a need for a basic level of functionality and this involves some repetition of previous developments, but beyond this the emphasis of these new tools should be on interchangeability: the power to import data from and export to existing packages which already accomplish some requirements. For example, built into the current version of Replaytool is a way of importing data from system log files (typically data written out by a computer which describes a process taking place, often in a very unfriendly format!) and converts this data for use in Replaytool. During this process the data is de-cluttered and made more human-readable, for example by replacing machine-oriented date and time records with human friendly ones, and replacing system ID codes with meaningful names. These new log formats can be exported to standard .txt files, with or without the added annotations inserted at the appropriate points.

The flexible ‘viewer’-oriented architecture of Replaytool allows supplied Java abstract viewer classes to be extended and tailored to create viewers for additional data types that require replay. This makes it very easy to build viewers for new formats of time series data. Some examples of common viewers which have been implemented to date are presented in the following section.

Example Replaytool viewers

This section presents some of the data viewers currently available in Replaytool, and therefore the file and data types currently directly supported.

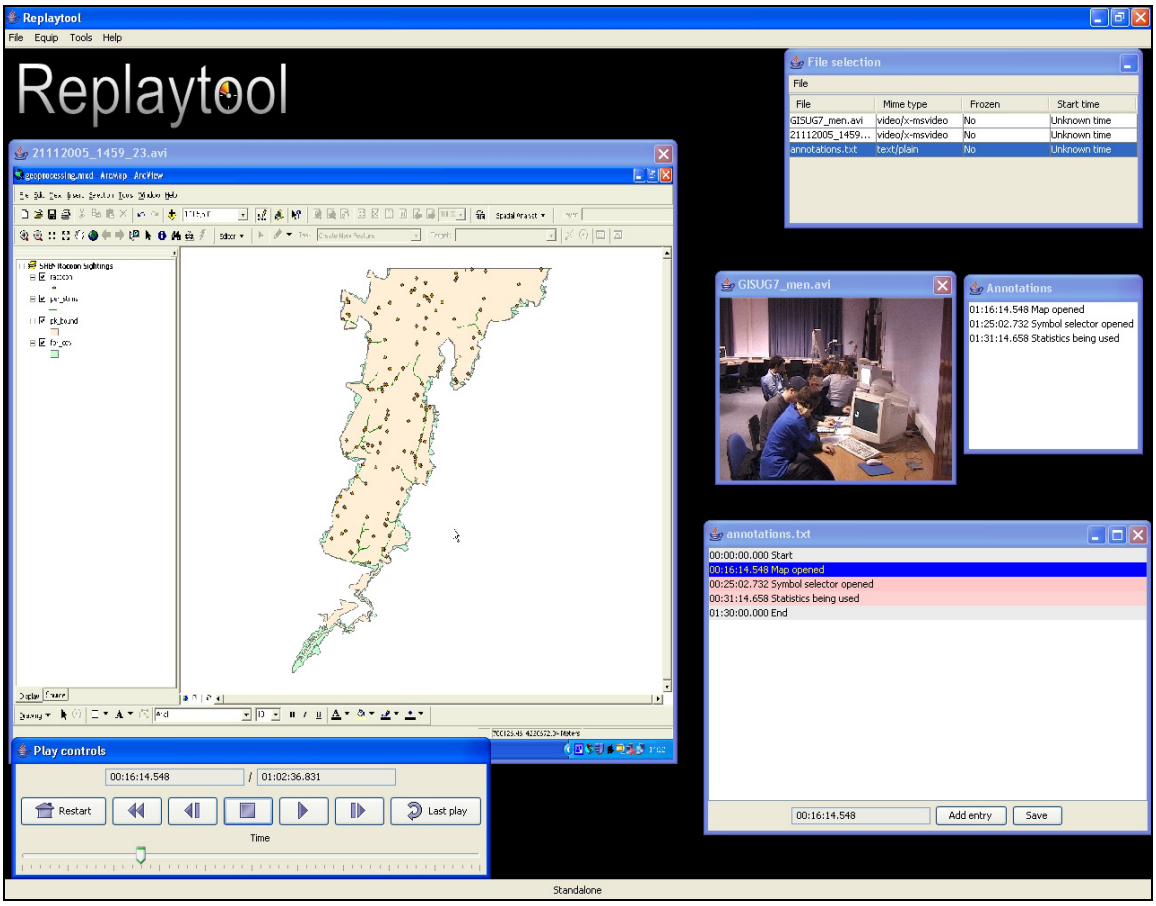


Figure 5. Replaying two videos simultaneously, in this case a screen capture file and a video of a group operating the computer. Annotations have been added by the user.

Replaytool is capable of displaying as many video viewers as system resources will allow. Each video can be independently synchronized. This allows the analyst to replay as much video data as he or she wishes, limited mainly by the viewable size on the monitors available. The technology behind the original Replaytool has been improved, and support has been added to replay Apple Quicktime-format movies.

As well as video files, system logs are another common format required for playback. An example of a viewer for this type of data can be seen in Figure 6:

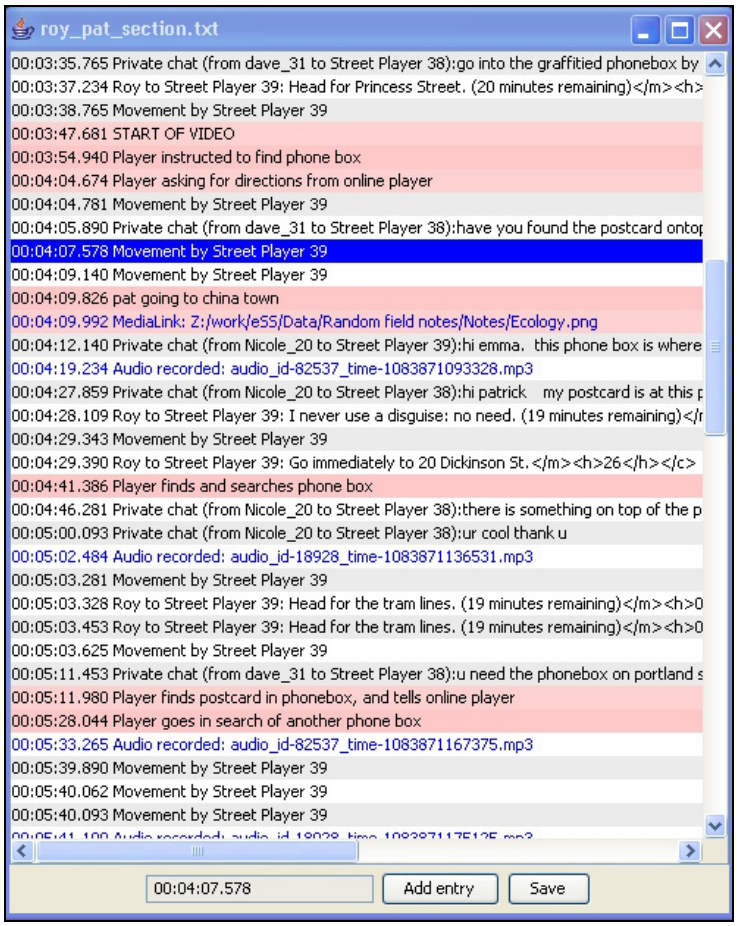


Figure 6. Imported system log file with interleaved annotations drawn from a database store.

This viewer displays lines of time-stamped log files. The user is able to click on a line to shuttle the current playback point to the time that line was recorded. Annotations can be 'inserted' into this viewer at appropriate points. In fact, these annotations are stored directly in the backend relational database – the original log file is unaffected. When the log file is loaded into this viewer, the system checks the database for any annotations, and automatically adds these lines to the viewer's version of the log file. It is possible to save this newly compiled file as a text file if the user requires.

As playback progresses, the most recent line of the log file is highlighted, giving the user clear feedback as to which item in the log the current time relates to, if any.

This viewer also allows the user to manipulate the ordering of the log files and annotations, using features emulating the cut and paste features of word processors. This was found to aid the re-working of a record into an interactional order, sometimes required in ethnographic study (Crabtree, 2006)[1].

---

1 For more detailed information on the manual aspects of working with system logs, see Crabtree et al. *Working with Digital Records: Developing Tool Support* in these proceedings

Some data sets contain positional data. This can be imported into the relational database and replayed using the spatial viewer, as is Figure 7:

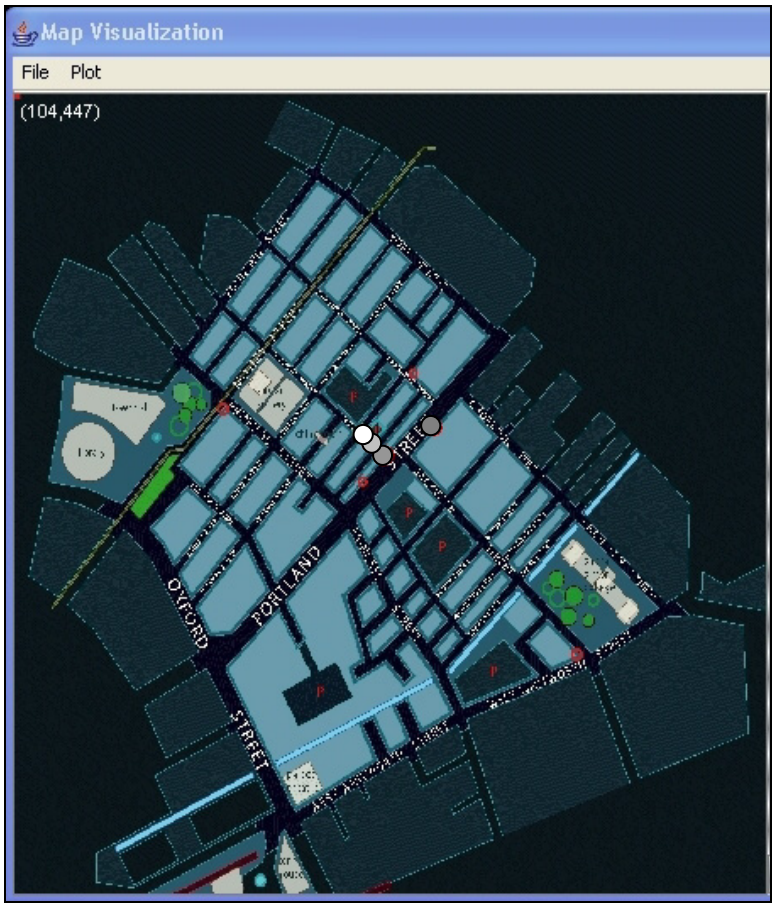


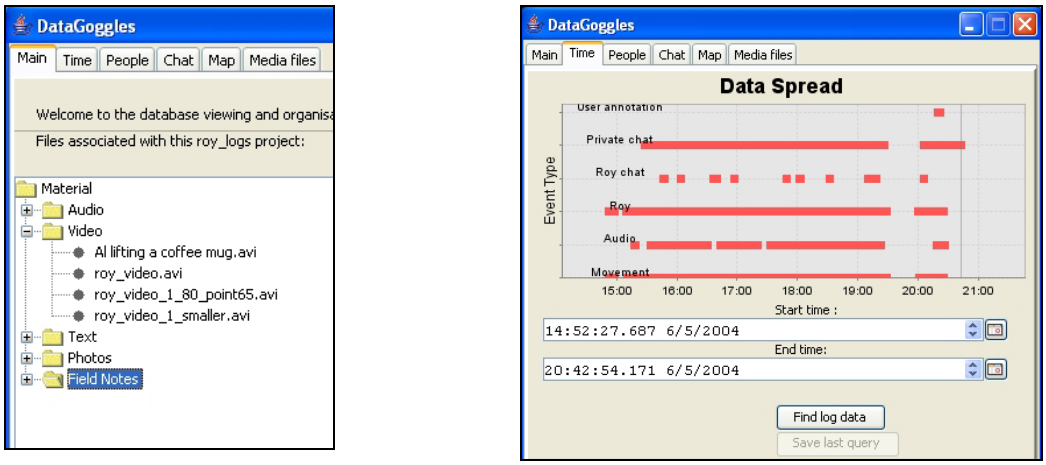
Figure 7. Coloured dots representing positions can either fade in and fade out as their particular time passes, or can be connected with lines giving a visualization of the path.

The map viewer can be used to display positions over time, and can also be used to shuttle the time based on activities which occurred at certain times in certain areas.

Any combination of viewers can be on screen at once, allowing the user to compare data from different sources at certain times, and allowing a replay of all the data available in real time. These viewers are developed and 'plugged in' to Replaytool in a modular fashion allowing easy future extensibility by creating specific viewers for new data types. For example, a custom viewer was recently written to allow the replay of log files from the EQUATOR IRC's EQUIP (Greenhalgh, 2002) system.

Visualization and management of data

As well as the replay and annotation of the data, it is important to manage the processes prior to this – namely the collection and organization of files, and the synchronization of data sources. Replaytool contains a prototype tool for this purpose, called DataGoggles. This tool contains three main elements: project management, dataset visualization, and synchronization.



(a) (b)

```
00:00:00.000 START OF TIME WINDOW
05:13:00.734 Nicole_20 hiu emma
05:13:14.016 Nicole_20 i ve found my postcard
05:13:30.734 Nicole_20 can you help me fin uncle roy
05:19:22.891 Nicole_20 hi emma
05:19:35.000 Nicole_20 look on top of the phone booth
05:19:50.078 Nicole_20 thank you for following me
05:20:04.406 Nicole_20 this is my postcard
05:20:38.688 Nicole_20 i cannot seem to hearsee anything you say if
05:21:32.406 Nicole_20 if u are talking to me     maybe im missing sumpt
```

(c)

Figure 8. DataGoggles tool examples. (a) Example of media management tool. (b) Overview visualization of data imported from a system log file. (c) An extract of speech from a participant, using the chat extract tool.

The DataGoggles tool supports simple project management by allowing a user to create multiple distinct projects, and to keep track of files (such as media files and system logs) within these projects. Figure 8(a) shows some of the files currently associated with a particular analysis project.

Visualization of the data at this preliminary stage was found to be most useful when in the form of an overview. For example, Figure 8(b) above shows at a glance what kinds of log file messages are available at which times. This particular data is taken from a log file of a distributed mobile game, and an analyst can clearly see from this when people were playing the game, and what types of messages were being sent and received. This kind of seemingly trivial data understanding is actually very hard to get from the raw data in a typical computer-generated log file, yet it is invaluable to the analyst to provide a way of getting into the data itself. Other types of visualization at this stage include maps with spatial positions plotted over a whole session (providing an overview of which areas were visited, for example), representations of the data files available for replay, and ways of extracting conversation from the log files between certain people and time constraints, eg. Figure 8(c).

By way of a representative example, consider an analyst using the Replaytool and DataGoggles software. In this example, the analyst is assumed to have a video file and a corresponding system log file which they wish the replay and annotate. First, the analyst adds the video file to the current analysis project. Next the log file's format must be made to match

that accepted by Replaytool and the logged events loaded into DataGoggles own event database. At present Replaytool by default accepts log files in the format:

```
TIMESTAMP DATA1, DATA2, DATA3, …
```

where `TIMESTAMP` is represented as hours:minutes:seconds.milliseconds relative to the first event in the file, which should have a time of `00:00:00.000`. Custom import filters can be written for DataGoggles, which can generate this kind of timestamp from other timestamp formats, such as the UNIX-style timestamps present in many system log files. With some files, DataGoggles can also automatically determine the real start time of the log file, to aid synchronization.

To do the actual replay and annotation, the analyst now exports the events and media files of interest from DataGoggles to create a Replaytool "fileset", which describes to Replaytool the set of files which are to be used together in a replay session and their time-offsets. Using DataGoggles, the user may now add multiple video files to this fileset, and may add a known real-world start time and date with those files (this information allows Replaytool synchronization information to be automatically generated). The analyst also selects the events of interest, for example from the overview visualization in figure 8(b). Finally, within Replaytool itself the fileset can be opened, allowing the video file to be loaded into a video file viewer and the log file into a log file viewer so that the two data sources can be replayed concurrently. At this point, the user is free to replay and annotate the data set as s/he wishes, and can fine tune the synchronization if it is not correct.

Current and Future Development

The first iteration of Replaytool, described in the main body of this paper, has achieved two goals. It has established a foundation level software framework on which to build and test future extensions, and it has allowed the software and the approaches which it embodies to be piloted with users and preliminarily tested. The preliminary feasibility testing has involved the three driver projects of the DReSS node: ethnographic analysis of ubiquitous computing applications, video-based language corpus studies and combined qualitative-quantitative studies of learning[2]. This testing has highlighted a number of areas which need addressing to make the tool more usable to the social science community.

First, there have been a number of shortcomings highlighted with the current annotation system of time-point free-text entries. This system is flexible, allowing any textual content such as descriptions or hyperlinks, but currently only *points in time* can be annotated. This means that there a number of useful annotations which are hard to make, including annotating *sections* of video, and annotating resources themselves (e.g. adding metadata about a file). The ability to apply structured annotations, such as coding schemes, will be built into the tool in the next major iteration (currently in progress). Additionally, the ability to annotate sections of sequenced, non-temporal material (such as text extracts or transcriptions) is beyond the capability of the current system. The solutions to these requirements is being implemented based around a new RDF (Resource Description Framework, (W3C, 2006a)) data model, with some other semantic web-type technologies to permit a very flexible and detailed representation of metadata and annotations. We have developed a number of ontologies using the Web Ontology Language (W3C, 2006b), one such semantic web technology, to describe

---

2 Example data replay from these three driver projects can be seen on the poster "'Replaytool' software in practice" at this conference.

the knowledge structure required for storing annotations and other such metadata. This ontology is designed to allow the annotation of any part of a project, from a file (e.g. metadata about who recorded a video, who has altered it, etc.), to annotation of sequences (implementing the relevant parts of the annotation graph method (Bird and Lieberman, 2001)) and also allowing the construction of coding schemes.

The RDF store will be managed using Jena (JENA, 2006), an open source Java-based framework for managing RDF stores, in our case over a relational database, MySQL. A server-client paradigm will be used, where the RDF data will be stored on a central (workgroup) web-service fronted database, and relevant sections of it can be downloaded to a local database on the user's machine when needed. The user makes their changes locally, and the updated RDF is then returned to the central database. This model permits a 'shared' central resource, allowing multiple people to work on a data set at any one time, whilst preserving the need for users to be able to work on machines which do not always have network connectivity. It also makes it much easier to manage and ensure the consistency of backups. As for raw data, users may choose to keep copies of media and data files (such as videos), local, or to allow them to be downloaded from the server. Although security will be implemented no system is ever completely secure, and this approach allows data with particularly stringent use restrictions to be used in a traditional (local only) manner, without the necessity to upload to a web server; this local data model ensures access to the data lies squarely with the researcher.

This new data management approach will also allow the system to represent an observation-driven perspective on the data. What this means is that time-specific items such as annotations, video start times and log file entries will be allocated a source of their timecodes – e.g. a camcorder clock, a server timebase etc. It will then be possible to offset and relate these independent timebases against each other, making synchronization of data easier. Also, researchers will be able to reconstruct their own perspective of the order of events without having to alter the source data or main synchronization settings: basically, a user will be able to create different interactional orders of events from different perspectives, e.g. one perspective might be a server-oriented view of when log events were recorded, and a second perspective might be a researcher-oriented view of when these events actually happened or were used in the human interaction (often not the same).

Synchronizing multiple data sources has always been a challenge, some specific examples of which were raised by the prototype software. One phenomenon is that of 'sync slip', where synchronization drifts over time. This is often caused by recording devices such as camcorders recording data at a slightly variable rate, so when two different sources are played back against a very accurate timeline, the two sources often drift over time. This phenomenon has a typically small effect, often in the order of a few seconds over many hours. However, it is worth considering the possible ways of correcting this for future versions of the software, as this is a problem across the board for social science research, and for work of a required high granularity in time, this effect must be addressed. With the new ontology, meta-information may be stored about devices. Therefore, if a device, such as an analogue tape recorded, is known to stretch recordings (for example if one hours recording actually takes one hour and one minute to playback), then this information can be stored for future use by corrective algorithms. For example, rather than assuming a single constant time offset between files, the new system will allow a sequence of time-dependent offsets to be specified, allowing the time offset to be continually adjusted during reply based on multiple observed moments of coincidence.

Current development towards a track-style view of the media sources is in progress. This will aid the user's conceptualization of the synchronizing process, providing an intuitive method to shift media sources relative to each other. This viewer would also provide another medium for displaying annotations.

Importing and exporting data allows a software tool to be used in conjunction with existing tools, allowing users to use a specific existing tool they are familiar with to do a specific job. Replaytool is planned to support some common standards for data, such as comma-separated text files which allow the interchange of data with spreadsheet programs. It is planned that it will be possible to make and export video clips in certain formats for use in presentations etc.

Lastly, but importantly, work is underway to make a release version of Replaytool suitable for more general distribution, for external user testing. This is hoped to happen shortly. Contact the authors for more information.

Discussion

A description of the Replaytool software developed by the DReSS NCeSS node has been presented. The motivation behind this development has been to provide a tool, targeted for use by social scientists, to provide a way of replaying and annotating various media sources in a flexible way. The first prototype of this software shows promise and has raised a number of significant issues which are being further addressed in the next version of this tool, which will feature a much more flexible ontology-based data model.

Acknowledgements

The authors would like to acknowledge work on prior versions of the software during the VidGrid project, and from Stuart Reeves. Thanks also for the input from everyone at the DReSS node, and for the support from the MiMeG NCeSS node.

References

Bird, S. and Liberman, M. (2001): 'A Formal Framework for Linguistic Annotation', *Speech Communication*, vol. 33, no.s 1-2, January 2001, pp. 23-60.

Crabtree, A, French, A., Greenhalgh, C. et al. (2006). 'Developing Digital Records: Early Experiences of Record and Replay'. *To appear in Computer Supported Cooperative Work: The Journal of Collaborative Computing, Special Issue on e-Research.*

Greenhalgh, C. (2002): 'EQUIP: a Software Platform for Distributed Interactive Systems', *Equator IRC Technical Report Equator-02-002.*

JENA (2006): 'A Semantic Web Framework for Java', http://jena.sourceforge.net/index.html, accessed 3rd May 05.

Saferstein, B. (2004): 'Digital technology and methodological adaptation'. *Journal of Applied Linguistics*, vol. 1.2, 2004, pp.197-223.

W3C (2006a): 'Resource Description Framework (RDF)'. http://www.w3.org/RDF/, accessed 3rd May 2006

W3C (2006b): 'Web Ontology Language (OWL)'. http://www.w3.org/2004/OWL/, accessed 3rd May 2006