From ReplayTool to Digital Replay System

Chris Greenhalgh, Andy French, Paul Tennent, Jan Humble, Andy Crabtree

School of Computer Science, University of Nottingham, UK.

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

Abstract. DRS, the Digital Replay System, is a software tool being developed by the DReSS node of the UK ESRC-funded National Centre for e-Social Science. It has been developed from the previous ReplayTool application to support the coordinated replay, annotation and analysis of combinations of video, audio, transcripts, images and system log files. DRS uses a new internal data model which gives it much greater flexibility than ReplayTool. It also provides new facilities for project and data management and organization, complex synchronization between related media, structured annotation including transcription and coding (classification), and new support for processing and visualizing log files and databases. It is publicly available under an open source license and is hosted by SourceForge. The current (first public) release emphasizes usability with a core feature set. Two further releases are planned which will make more experimental facilities available to general users.

Introduction

ReplayTool (French et al., 2006) was the initial prototype developed by the NCeSS DReSS Research Node[1] of a suite of tools to enable social scientists to handle 'digital records' (Crabtree et al., 2006b). Digital records consist of two essential components: 1) traditional resources that social scientists working in qualitative traditions might gather (video and/or audio recordings, transcripts, photographs, etc.) and 2) 'system logs' or electronic recordings of events including interaction in computational environments. DRS and ReplayTool enable time-based data – i.e., system recordings and audio/visual recordings – to be combined and replayed side-by-side and for annotations to be added to create new representations.

As described by Crabtree et el. (2006b) ReplayTool was used and extended to support ethnographic analysis of the "Uncle Roy" mobile game/experience as a driving application. In addition the regular meetings of the DReSS node, which includes members from the Schools of Psychology and English, have also been used to share and explore analytical practice and requirements across a range of settings and perspectives. This has led to the identification and prioritization of further requirements. In the second major phase of development activity within DReSS we are responding to these through a major re-engineering and extension of ReplayTool to create the "Digital Replay System" (DRS). This paper describes how the following requirements have been addressed:

- **Generalized support for project and data management and data overview**: the "DataGoggles" component described by French et al. (2006) was hand-tailored for a particular pilot project and demonstration purposes.

[1] http://www.ncess.ac.uk/research/nodes/DigitalRecord/

- **Complex synchronization between multiple related media and log files**: e.g., if different analysts have different views of the best correspondence between media or if video recordings run at different speeds or are discontinuous.

- **Complex and structured annotations**: e.g., of time intervals and of non-temporal extents, and annotations which consist of structured codes rather than just text.

- **Support for log-file processing, storage and visualization within the tool-supported environment**: e.g., to support repeatability, sharing and re-use of such elements.

In this paper we consider each requirement in turn, describing how they are being addressed in DRS. We then give some examples of current use, and explain how to obtain DRS. The next section briefly describes the main technical changes in data modeling and persistence that underlie the other enhancements that follow.

Internal Data Modeling and Persistence

To address these diverse requirements the internal data model and storage mechanism for DRS has been changed from a simple file-based XML data model for viewing a single set of media files in ReplayTool to a more comprehensive and extensible data and metadata model based on the W3C's Resource Description Framework (RDF) and Web Ontology Language (OWL), both Semantic Web technologies supported by the Open Source JENA RDF library for Java.[2] The DRS ontology has been created using the Stanford Protégé Ontology editor,[3] and includes portions for:

- Its own configuration, e.g., workgroup or standalone, system users, JENA models and window layouts.

- The files and databases that it is managing.

- Other information that the system explicitly uses and depends on, e.g., projects, analyses, timing and synchronization, codes and annotations.

- Other metadata associated with any of these, e.g., participants in studies, devices used, etc.

This has given us a flexible platform for description and persistence within DRS which in turn has allowed us to respond to the above requirements (see Greenhalgh et al. (2007) for more technical details on the use of RDF and OWL within DRS).

Projects and "analyses"

The DataGoggles component of ReplayTool (French et al., 2006) provided an example of a project overview and the facility to export events and combinations of media to a simple ReplayTool session. However, this capability was significantly hand-tailored for use with the Uncle Roy driver application. DRS now provides a general "Project" mechanism whereby multiple files and annotation sets can be managed and viewed as a set of distinct "Analyses". For each project a graphical Project Explorer (figure 1) provides a simple entry point to and

---

2 See http://www.w3.org/RDF and http://www.w3.org/TR/owl-features

3 See http://protege.stanford.edu

representation of the project's elements and organization (e.g., analyses, media, people). The popup (context) menus for the items within the browser give access to available operations for each item (e.g. to associate some media with a particular analysis).

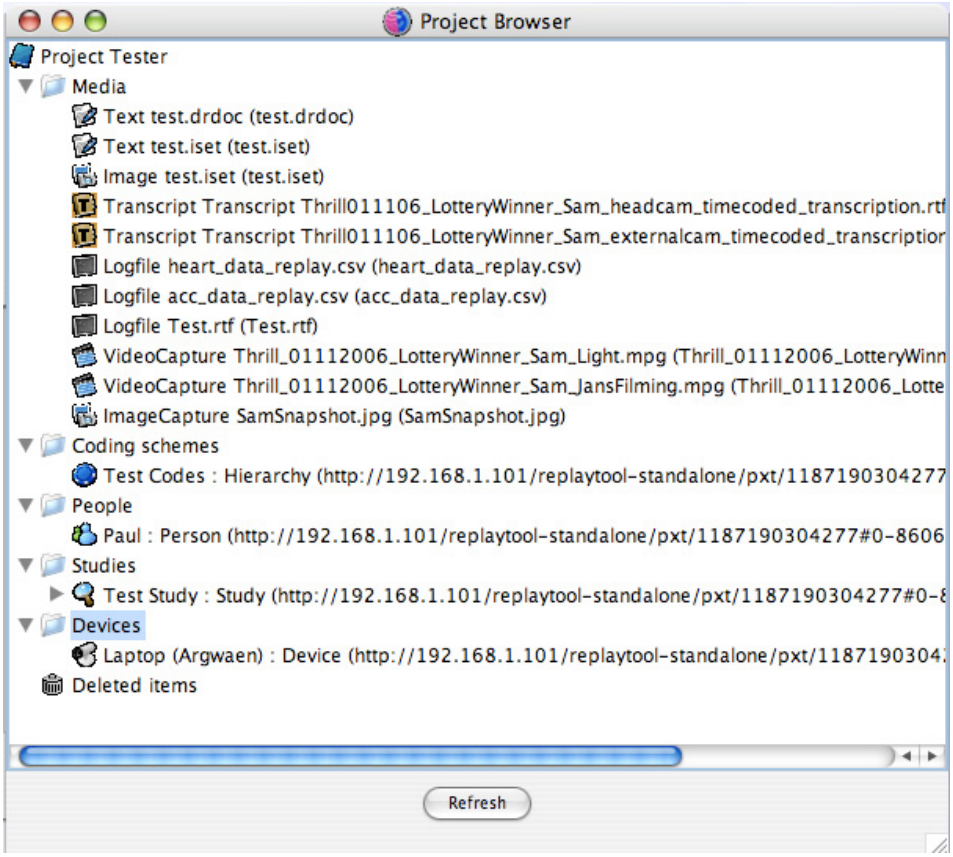

Figure 1. The project browser.

In the terminology of DRS an "analysis" is a set of related (generally co-temporal) resources, potentially including digitized videos and audio recordings, images, transcripts, annotations and log files. Each analysis is viewed and manipulated via a similar analysis browser which shows only the resources specifically associated with that analysis. In addition, play-back within each analysis is controlled via VCR-like controls (comparable to ReplayTool).

A time-line view is available for each analysis, giving a visual representation of the temporal extent and offsets of the media files, as well as visual representations of audio waveforms, coding and annotation (see figure 2). A basic concordance-style search view is also available to search across all annotations (coding and transcripts) within a project (see figure 3). From here the analyst can open and jump to the analysis and time associated with the text.

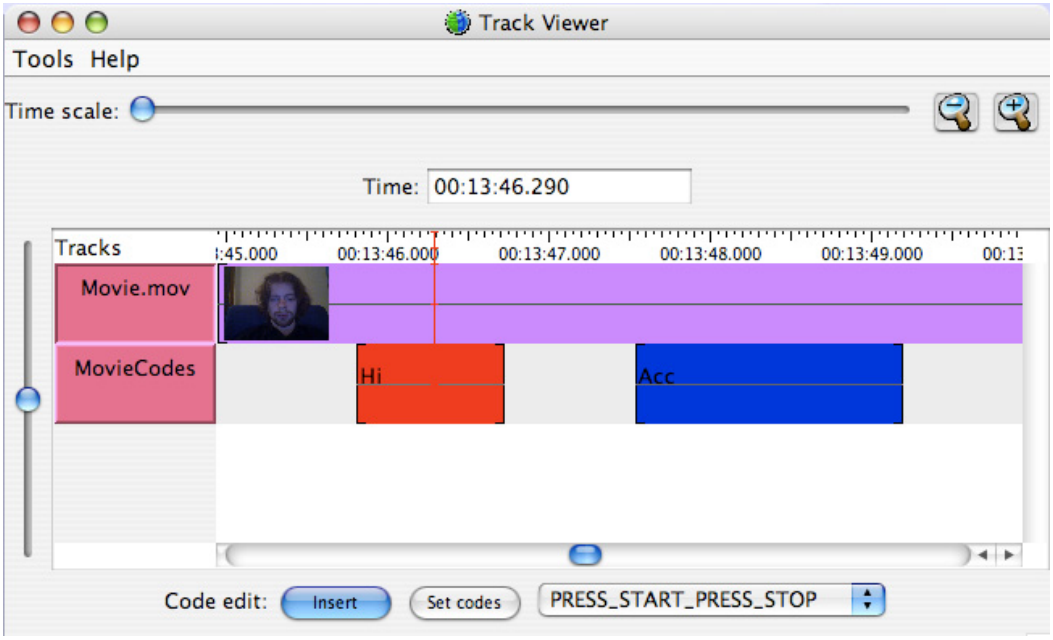

Figure 2. The track viewer showing the time-line for an analysis with one video and one annotation (coding) track.

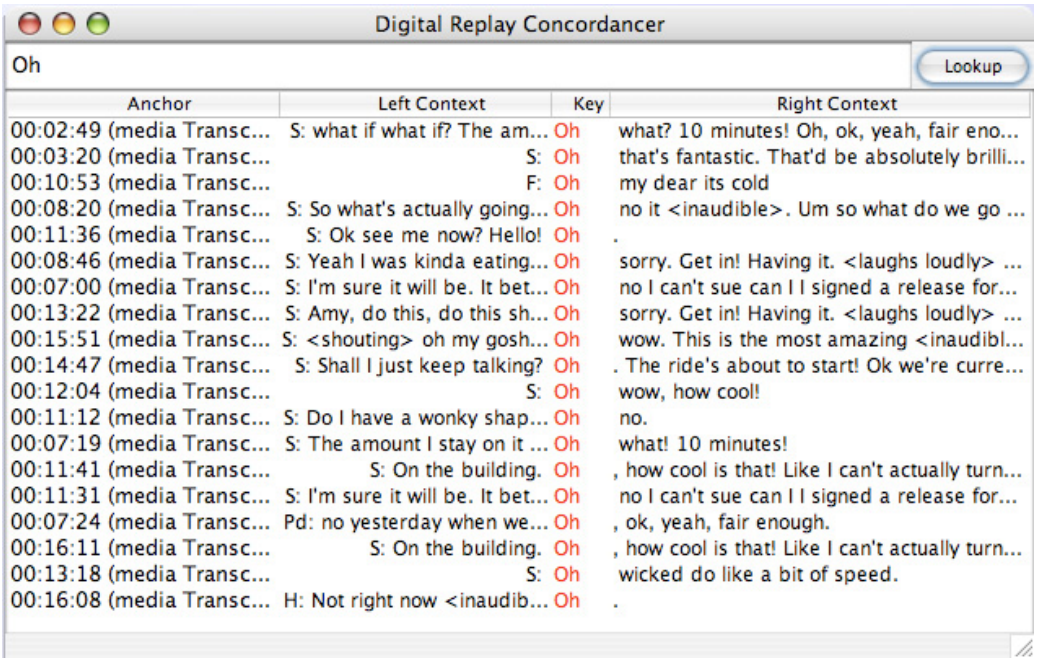

Figure 3. Searching in the concordance view
(the left column links to the particular media or analysis).

Synchronization

In terms of synchronization between related media and log files, ReplayTool had a simple model with a single viewing timeline and a single offset between that and each media/log file. In DRS every time-based media file and every analysis has its own abstract “timeline”. For example, the track viewer (above, figure 2), shows the temporal relationships between the

video ‘Movie.mov’, the coding track ‘MovieCodes’ and the current analysis. These temporal relationships can be expressed in several different ways:

- The movie and analysis may have explicit start date/times specified, from which their relationship is inferred – see figure 4.

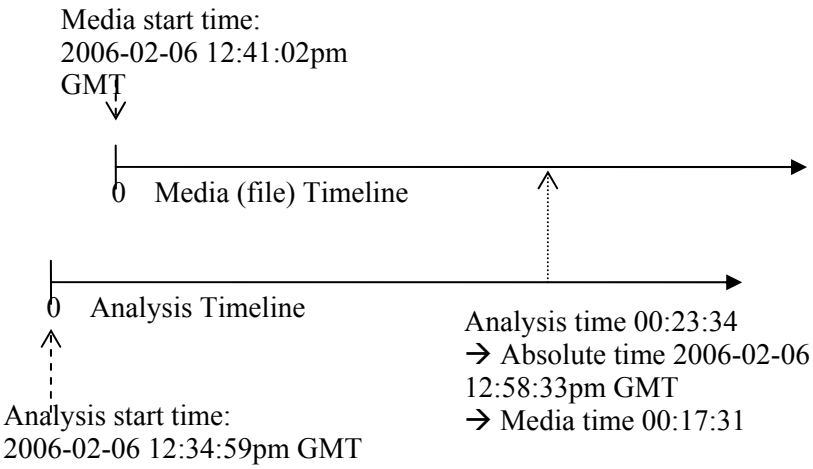

Figure 4. Synchronization using explicit dates and times.

- The movie and analysis may have a directly specified relationship between their own timelines independent of the absolute date/time – see figure 5. This temporal relationship can be different for different analyses (e.g. to represent different perspectives on the same event(s)).

- In addition to the above options, the coding track may be explicitly linked to the same timeline as the movie (or any other time-based media), e.g. if it is specifically a coding of what is happening in that other media.

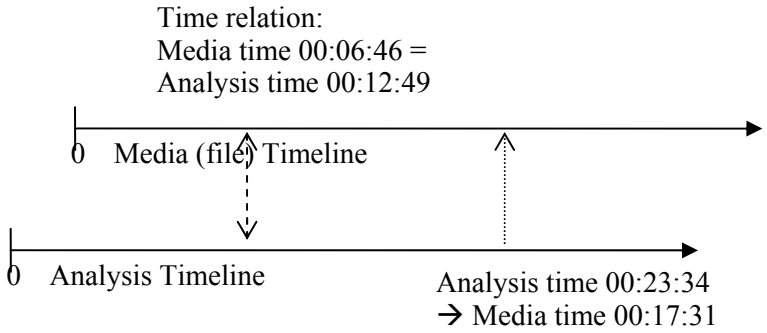

Figure 5. Synchronization using explicit timeline relationships.

Media can be synchronized using the track viewer (figure 2), by dragging individual tracks along the analysis time-line. More comprehensive synchronization options can be found in the Synchronization Manager window, including specifying explicit start times and temporal relationships.

Annotation

In terms of annotation ReplayTool supported only free-text annotations of single moments of viewing time. In the original ReplayTool these were simply time-stamped text lines in a textual log file and the DataGoggles component subsequently held them in a single common annotation table in the DataGoggles database.

Starting from the annotation graph approach of Bird and Liberman (2001) we defined in the DRS ontology (data model) a rich model of annotation as illustrated in figure 6. In general, each annotation associates some *subject* with some *content*. At present in DRS the main subject for an annotation is a *region*, in particular a region of time on the timeline of a piece of media or an analysis. Related annotations (e.g. of the same media) are organized together as *annotation sets*, which appear as a form of media with DRS (e.g. in the project and analysis browsers, as in figure 1).

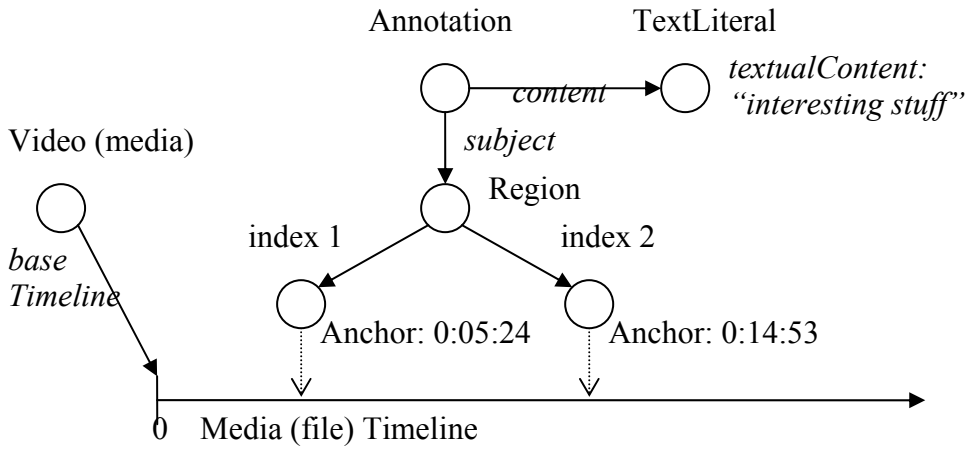

Figure 6. Annotation of video segment 0:05:24-0:14:53 with "interesting stuff" (omitting details of Anchor_index and RelativeTime)

DRS supports the creation and time-synchronised viewing of text transcripts alongside other media. A transcript is stored internally as a (temporally ordered) set of non-overlapping free-text annotations. This can also be converted to and from an RTF (Rich Text Format) file compatible with the Transana transcription and analysis tool[4]. Currently transcripts can be viewed and edited as a (virtual) document (figure 7, left) or a track in the track viewer or viewed as a table of annotations (figure 7, right).

Annotations can also have content which is not simply text. In particular, the analyst can create their own *coding schemes*, which typically consist of a hierarchy of *codes* (effectively keywords). These codes can then be used as the content of an annotation to allow more structured coding (classification) of observations. Coding schemes are viewed and edited using a simple tree-based view. A coding scheme can have one of several different temporal characters: event with nominal duration (i.e. duration is not explicitly specified); event with variable duration; switching state (i.e. one code must always apply, and only changes are specified); or periodic state (i.e. one code must always apply, and it is identified at fixed intervals).

4 http://www.transana.org/

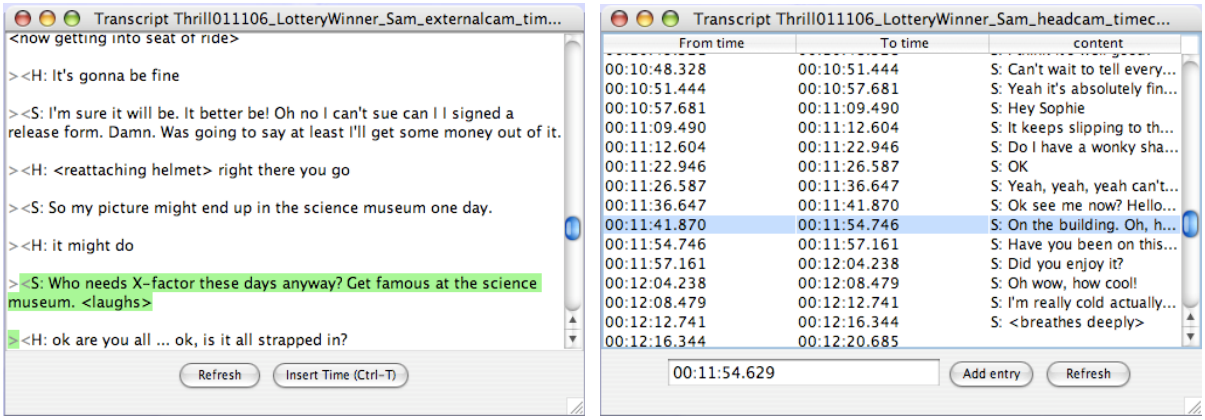

Figure 7. Virtual document and annotation views of a transcript.

A *code track* is an annotation set associated with a particular coding scheme, i.e. only the codes from that coding scheme can be used as the content of its annotations. Coding is currently done exclusively using the track viewer (figure 2) in the particular code track. The process of coding is quite flexible, allowing coding using keyboard (user-specified short-cuts) or mouse, and allowing segmentation (timing) to be done before or at the same time as coding. It also takes account of the temporal character of the coding scheme, e.g. durations (stop times) are only required for variable duration event coding schemes. In DRS the coding scheme can be modified during the coding process, e.g. to reflect emerging analytical refinements.

Working with log files

Within ReplayTool, log files had to be converted to a standard time-stamped text format before they could be viewed. In the DataGoggles component the internal format became relational tables, and the conversion process was partially automated in that files could be imported via the DataGoggles interface. However, the DataGoggles component required hand-specification of the database schema for different kinds of log information, hand-crafted file import code, and left the original log files outside of the normal file management facilities of the application.

For DRS a new plug-in interface has been defined which makes it simple for a Java programmer with little knowledge of DRS to write a procedure for processing a log or other computational file to generate intermediate data or a visual representation (a "view"). This is supported by the facility to dynamically create and manage multiple relational databases within a single DRS project, so that log data can be stored, manipulated and queried efficiently, inspired in part by the Replayer (sic) visualization and replay tool (Crabtree et al., 2006a).

A log file "workbench" interface within DRS (figure 9) allows users (rather than programmers) to configure and manage these log file processors and viewers, for example to read a particular log file, load its information into a dynamically-generated database, and then provide a particular view on the log from that database. The views can then be accessed directly from the main part of the DRS interface and can be synchronised with the normal replay of media files.

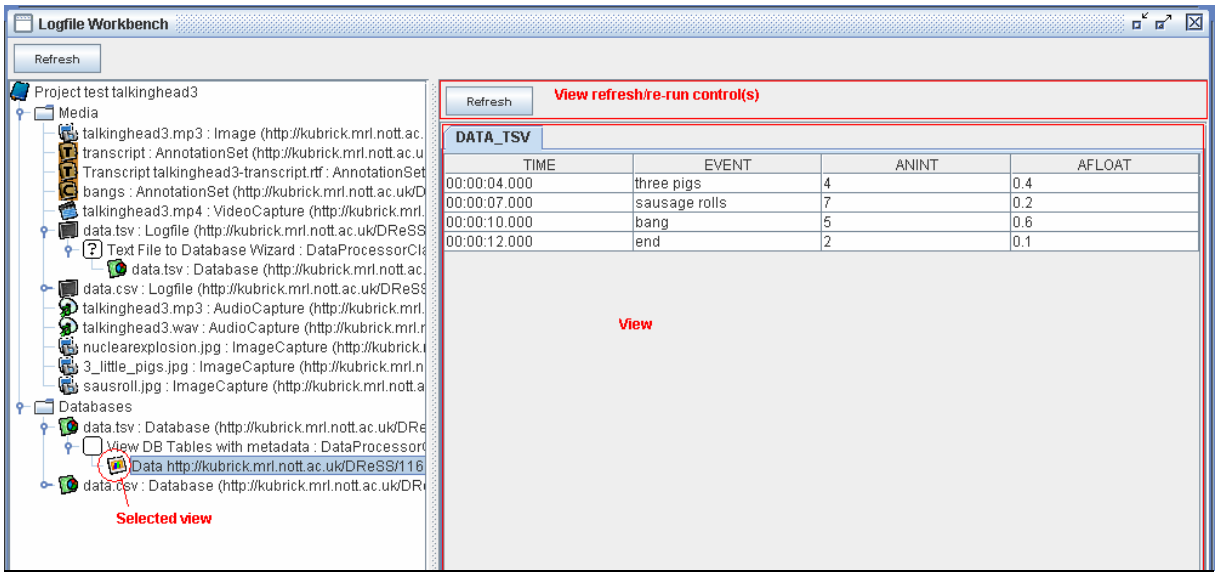

Figure 9. Logfile workbench, showing a generic database table view (right).

The current version of DRS includes a standard processor for converting a range of tabular text logs to a DRS database, e.g. comma-separated value files from Excel, SPSS or R. It also has generic table and simple graphing views available once the database has been created. We are continuing to extend the range of general-purpose importers and viewers, but in more specialized cases (e.g. proprietary log file formats or specific views) a custom processor can be written and added to DRS (by a Java programmer). For example, we have created custom database loaders and visualizations to support analysis of the Day of the Figurines mobile phone-based game/experience.

Examples of use

The first general release of DRS was made available on SourceForge on the 29th August 2007. Consequently examples of use to date are generally internal to the node, largely as part of the development described to clarify requirements and refine the user interface. These have included: replay and visualization of player activity logs from the Day of the Figurines game/experience; simple coding of gestures (head nods) in the associated HeadNod ESRC small grant project; review and preliminary coding of collaborative sessions using a VR training application by our node colleagues in Psychology; replay and searching of video recordings of two-party conversations (supervision meetings) by our node colleagues in English. We have also demonstrated DRS to a number of other groups and meetings to further explore potential use and requirements.

The most representative and comprehensive example of use to date is in the analysis of the experience of volunteers on theme park rides as part of the Fairground: Thrill Laboratory project commissioned by the London Science Museum's DANA Centre (Walker et al., 2007). Figure 10 shows DRS in use to replay a typical ride experience. Visible in the figure are the track viewer, showing an overview of the resources in play, and time-synchronized views of a text transcript, heart rate and acceleration data from sensors, two concurrent video recordings and the analysis playback controls.

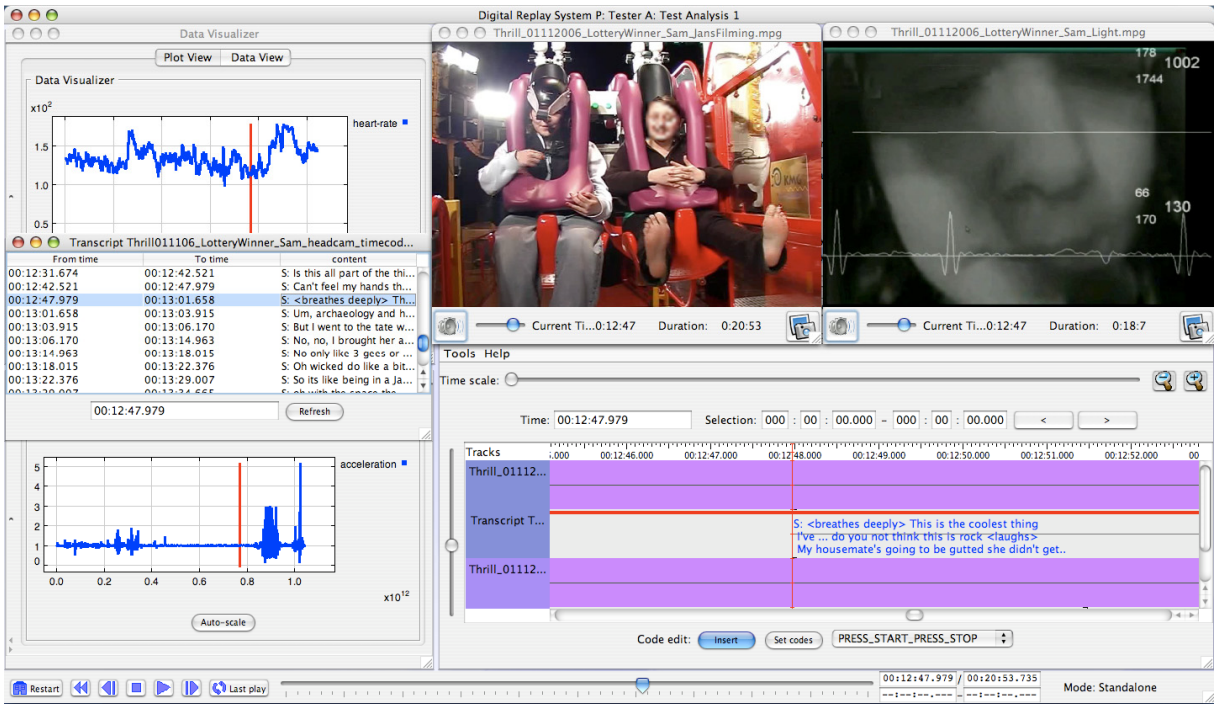

Figure 10. Using DRS to analyze the experience of a theme park ride in the Thrill project: track viewer (bottom right), text transcript (centre left), sensor data graphs (top and bottom left), video recordings (top right) and playback controls (bottom)

Availability

DRS is publicly available under the "new" BSD open source licence. It is hosted by SourceForge as http://thedrs.sourceforge.net. It is currently available to end-users as a Java WebStart application, linked from the project's Download page (it requires a broadband internet connection for the initial download and can then be used with or without an internet connection). It is currently being developed and tested on Windows (XP) and Mac (OSX), and requires Apple's (free) QuickTime system.

Conclusions and future work

For the first release of DRS (August 2007) we have concentrated on the usability of a core set of features and capabilities. As well as regular updates, two further releases (December 2007 & March 2008) are planned before the end of the current DReSS node funding, which will significantly extend the capability of the system. Some of the areas of planned development are noted below.

- The current version of DRS has a number of additional interfaces which allow project-related metadata to be recorded and viewed, e.g., relating media files to data capture sessions and participants. However these are only offered to "expert" users as they currently require a significant understanding of the underlying RDF metadata model. We plan to make a simpler subset of these facilities accessible to all users.

- The internal DRS data model allows drift and discontinuities between media files to be modeled as multiple local correspondences between their timelines. We plan to make this facility accessible from the user interface and supported by the internal viewers.

- From our requirements gathering we know that some examples of coding require, in addition to the codes themselves some mechanism for parameterization (e.g. to identify the specific object of a gesture or a measure of confidence). We have prototyped a system of *modifiers* which can be associated with codes, and we will make this facility generally available.

- As noted above we plan to release more reusable log file processors and viewers, for example to support map-based visualizations of position and movement.

- The first release of DRS is intended for standalone (individual) use. There is also a DRS workgroup server, which allows groups of researchers to collaborate asynchronously on the same project(s). This will be supported in the final release of DRS.

Acknowledgments

This work was supported by the ESRC through the grant "Understanding New Forms of Digital Record for E-Social Science" (the DReSS node of the NCeSS) and by the EPSRC through grant EP/C010078/1, "Semantic Media - Pervasive Annotation for e-Research" and the EQUATOR IRC, grant GR/N15986/01. We would like to thank the other members of the DReSS node for their collaboration, and the Thrill project for their pioneering use of DRS.

References

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

Crabtree, A., Benford, S., Greenhalgh, C., Tennent, P., Chalmers, M., and Brown, B. (2006a): 'Supporting ethnographic studies of ubiquitous computing in the wild', in *Proceedings of the 6th ACM Conference on Designing interactive Systems* (University Park, PA, USA, June 26 - 28, 2006). DIS '06. ACM Press, New York, NY, 60-69.

Crabtree, A., French, A., Greenhalgh, C., Rodden, T. and Benford, S. (2006b): 'Working with digital records: developing tool support', paper presented at the *2nd International Conference on e-Social Science*, June 28-30, Manchester: ESRC. www.ncess.ac.uk/events/conference/2006/papers/papers/CrabtreeDigitalRecords.pdf

French, A., Greenhalgh, C., Crabtree, A., Wright, M., Brundell, P., Hampshire, A., and Rodden, T. (2006): 'Software replay tools for time-based social science data', paper presented at the *2nd International Conference on e-Social Science*, 28-30 June 2006, Manchester, UK www.ncess.ac.uk/events/conference/2006/papers/papers/FrenchSoftwareReplayTools.pdf

Greenhalgh, C., French, A., Humble, J. and Tennent, P. (2007): 'Engineering a replay application based on RDF and OWL', *UK e-Science All Hands Meeting 2007*.

Walker, B., Schnädelbach, H., Egglestone, S. R., Clark, A., Orbach, T., Wright, M., Ng, K. H., French, A., Rodden, T., and Benford, S. (2007): 'Augmenting amusement rides with telemetry', In *Proceedings of the international Conference on Advances in Computer Entertainment Technology* (Salzburg, Austria, June 13 - 15, 2007). ACE '07, vol. 203. ACM Press, New York, NY, 115-122.