



University  
of Glasgow

Tennent, Paul R (2014) *Augmented analyses: supporting the study of ubiquitous computing systems*. PhD thesis.

<http://theses.gla.ac.uk/5307/>

Copyright and moral rights for this thesis are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the Author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the Author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

# Augmented Analyses

Supporting the Study of Ubiquitous Computing Systems

Paul Tennent

Department of Computing Science  
University of Glasgow



Thesis submitted for the Degree of Doctor of Philosophy  
in Computing Science

· June 2014 ·

## Abstract

Ubiquitous computing is becoming an increasingly prevalent part of our everyday lives. The reliance of society upon such devices as mobile phones, coupled with the increasing complexity of those devices is an example of how our everyday human-human interaction is affected by this phenomenon. Social scientists studying human-human interaction must now take into account the effects of these technologies not just on the interaction itself, but also on the approach required to study it. User evaluation is a challenging topic in ubiquitous computing. It is generally considered to be difficult, certainly more so than in previous computational settings. Heterogeneity in design, distributed and mobile users, invisible sensing systems and so on, all add up to render traditional methods of observation and evaluation insufficient to construct a complete view of interactional activity. These challenges necessitate the development of new observational technologies. This thesis explores some of those challenges and demonstrates that system logs, with suitable methods of synchronising, filtering and visualising them for use in conjunction with more traditional observational approaches such as video, can be used to overcome many of these issues.

Through a review of both the literature of the field, and the state of the art of computer aided qualitative data analysis software (CAQDAS), a series of guidelines are constructed showing what would be required of a software toolkit to meet the challenges of studying ubiquitous computing systems. It outlines the design and implementation of two such software packages, *Replayer* and *Digital Replay System*, which approach the problem from different angles, the former being focussed on visualising and exploring the data in system logs and the latter focussing on supporting the methods used by social scientists to perform qualitative analyses.

The thesis shows through case studies how this technique can be applied to add significant value to the qualitative analysis of ubiquitous computing systems: how the coordination of system logs and other media can help us find information in the data that would otherwise be inaccessible; an ability to perform studies in locations/settings that would otherwise be impossible, or at least very difficult; and how creating accessible qualitative data analysis tools allows people to study particular settings or technologies who could not have studied them before. This software aims to demonstrate the direction in which other CAQDAS packages may have to move in order to support the study of the characteristics of human-computer and human-human interaction in a world increasingly reliant upon ubiquitous computing technology.

# Declaration

The work reported in this thesis has not been submitted in support of an application for another degree at this or any another university.

University of Glasgow, June 2014, Paul Tennent



# Acknowledgments

Bringing this thesis to fruition has been a long and torturous process for many people. Those people deserve extensive thanks, and that is exactly what they will receive here.

First and foremost I would like to thank Matthew Chalmers. Supervisor, occasional therapist, supporter, and all-round good guy. His undergraduate lectures on HCI sparked my interest in the subject and by the time I had completed his course I had decided that this academia lark was the thing for me. A certain amount of pestering later and I began the process of sauntering slowly towards this PhD.

Many other colleagues old and new require thanking here. The other PhD students with whom I shared my formative academic years, Marek Bell, Malcolm Hall and Scott Sherwood are singled out for particular thanks, with many other colleagues at Glasgow (particularly Barry Brown and Louise Barkhuus) providing me with much support, both academically and personally. Equally singled out for specific acknowledgement is Alistair Morrison who helped kick Replayer into life.

Moving away from Glasgow to Nottingham has the unfortunate effect of requiring me to thank lots of newer colleagues: Andy Crabtree, Tom Rodden, Chris Greenhalgh and Steve Benford as bosses of various levels - each of whom deserves specific thanks in their own right, Pat Brundell and Dawn Knight as fellow researchers in the dark days of DRS, and more recently Michel Valstar for being everything from academic mentor to best friend over the last year. Special thanks also go to Nadia Pantidi, Stuart Moran, Khaled Bachour, Richard Wetzell, Daniela Dybalova and Martin Flintham for distracting me with board games and generally making this take longer than it should have, and last but certainly not least of my colleagues: Brendan Walker - a man who has variously believed in me, befriended me, supported me, employed me, and entertained me over several years of thrill-oriented research.

And so we come to family. I am delighted to note that I have a family that understands when to offer support and (usually) when to shut up. My loving Parents, Robert and Dinah Tennent have stood by me throughout this process, casually looking over my shoulder and occasionally questioning my grammar, but mostly telling me everything was going to be okay.

Finally, Jennifer Bradbury. The most important person in my life.

---

This work was generously funded by the Equator IRC and through them by the EPSRC.

# Dedication

For my parents, Robert and Dinah Tennent. I wouldn't be here without them.

---

*"I love deadlines. I like the whooshing sound they make as they fly by." . . .*

Douglas Adams (1952-2001)

# Contents

<b>Declaration</b>	<b>i</b>
<b>Acknowledgments</b>	<b>ii</b>
<b>Dedication</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Approach Taken . . . . .	3
1.3 Contributions . . . . .	4
1.4 Thesis Statement . . . . .	5
<b>2 Related Work</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 Ubiquitous Computing . . . . .	6
2.2.1 The Origins of Ubiquitous Computing . . . . .	6
2.2.2 What do we really mean by a Ubicomp System? . . . . .	8
2.3 Evaluation . . . . .	11
2.3.1 Specific Challenges of Evaluating Ubiquitous Computing Systems . . . . .	15
2.4 State of the art in CAQDAS . . . . .	17
2.4.1 Replay . . . . .	18
2.4.2 Commercial Tools . . . . .	18
2.4.3 Academically Developed Tools . . . . .	20
2.4.4 Project Specific Replay Tools . . . . .	22
2.4.5 Capture and Replay of System Log Data . . . . .	22
2.5 Example real uses of CAQDAS . . . . .	23
2.6 Analysing Log files by hand . . . . .	25
2.7 Design guidelines for ubiquitous computing analysis tools . . . . .	26
<b>3 Replay Systems</b>	<b>29</b>
3.1 Introduction . . . . .	29
3.2 Replayer . . . . .	30
3.2.1 Requirements and Design Approach . . . . .	30
3.2.2 The Server . . . . .	33
3.2.3 Clients . . . . .	33
3.2.4 Log Data Viewers . . . . .	36
3.2.5 Statistical Analysis Tools . . . . .	42
3.2.6 Google Earth Bridge . . . . .	44
3.2.7 Multimedia Components . . . . .	45
3.2.8 Summary . . . . .	46
3.3 Digital Replay System (DRS) . . . . .	47
3.3.1 Requirements and Design Approach . . . . .	48
3.3.2 Corpus Management . . . . .	49
3.3.3 Time and Synchronisation . . . . .	50

3.3.4	Media Files . . . . .	50
3.3.5	Media Viewers . . . . .	51
3.3.6	The Track Viewer . . . . .	55
3.3.7	Transcriptions . . . . .	56
3.3.8	Coding . . . . .	58
3.3.9	The DRS Document Viewer . . . . .	61
3.3.10	Gesture Tracker . . . . .	61
3.3.11	The Concordance Viewer . . . . .	65
3.3.12	Log File Workbench . . . . .	67
3.3.13	Summary . . . . .	67
3.4	Conclusions . . . . .	68
<b>4</b>	<b>Key Implementation Factors</b>	<b>69</b>
4.1	Introduction . . . . .	69
4.2	Log Files . . . . .	69
4.2.1	Introduction . . . . .	69
4.2.2	State and Event Logging . . . . .	70
4.2.3	Replayer's Log File Handling . . . . .	71
4.2.4	Log Files and Parsers . . . . .	71
4.2.5	Instrumentation . . . . .	72
4.2.6	Log File Parsers . . . . .	74
4.2.7	Log File Handling in DRS . . . . .	76
4.2.8	Comparing the two approaches . . . . .	78
4.3	Distributed software architecture . . . . .	79
4.3.1	Introduction . . . . .	79
4.3.2	Replayer's Client-Server architecture . . . . .	80
4.3.3	The Server . . . . .	81
4.3.4	Communications Protocol . . . . .	82
4.3.5	The Database . . . . .	83
4.3.6	Queries . . . . .	85
4.3.7	Clients . . . . .	86
4.3.8	Bridge Components . . . . .	89
4.3.9	Extensibility . . . . .	93
4.3.10	Digital Replay System's Client-server system . . . . .	93
4.3.11	Server Overview . . . . .	94
4.3.12	Security and access control . . . . .	94
4.3.13	Client Access control . . . . .	96
4.3.14	Project Synchronization . . . . .	97
4.3.15	Synchronising client and server . . . . .	98
4.3.16	Comparing the two approaches . . . . .	102
4.4	Synchronization . . . . .	103
4.4.1	Introduction . . . . .	103
4.4.2	Synchronization in Replayer . . . . .	103
4.4.3	Synchronization in DRS . . . . .	107
4.4.4	Comparing the two approaches . . . . .	111
4.5	Conclusion . . . . .	112
<b>5</b>	<b>Case Studies</b>	<b>113</b>
5.1	Introduction . . . . .	113
5.2	Treasure . . . . .	113
5.2.1	Inaccurate Interface Displays . . . . .	116
5.2.2	Pick Pockets . . . . .	118
5.2.3	Environment Awareness . . . . .	122
5.2.4	Summary . . . . .	125
5.3	Day of the Figurines . . . . .	126

5.3.1	Summary . . . . .	134
5.4	Other examples of the use of Replayer and DRS . . . . .	134
5.5	Conclusions . . . . .	135
<b>6</b>	<b>Reflections and Conclusions</b>	<b>137</b>
6.1	Replayer vs DRS . . . . .	137
6.2	Revisiting the challenges . . . . .	142
6.2.1	Mobility . . . . .	142
6.2.2	Small Displays . . . . .	144
6.2.3	Headphones . . . . .	144
6.2.4	Non-Collocation . . . . .	145
6.2.5	Invisible Interaction . . . . .	146
6.2.6	Distribution of Interaction . . . . .	146
6.2.7	Interaction Time . . . . .	148
6.2.8	Technological Breakdowns . . . . .	149
6.2.9	Meeting the Challenges . . . . .	149
6.3	Revisiting the design guidelines . . . . .	150
6.4	Specific Academic Contributions . . . . .	151
6.5	Conclusion . . . . .	154
<b>A</b>	<b>Author's contribution to described software</b>	<b>182</b>
A.1	Replayer . . . . .	182
A.2	Digital Replay System . . . . .	183
<b>B</b>	<b>Addendum</b>	<b>184</b>
B.1	Introduction . . . . .	184
B.2	New Types of Data . . . . .	186
B.2.1	Social networking data . . . . .	186
B.2.2	Expression Recognition . . . . .	188
B.2.3	Physiological data . . . . .	191
B.3	Data processing . . . . .	193
B.4	New Systems . . . . .	195
B.5	Success stories . . . . .	197
B.6	Counter Examples . . . . .	200
B.7	Eliciting stories with data . . . . .	202
B.8	Reflection . . . . .	204
B.8.1	Revisiting the challenges again . . . . .	205
B.8.2	Conclusions . . . . .	207

# List of Figures

2.1	The Record and Reuse Process . . . . .	27
3.1	An early version of Replayer . . . . .	31
3.2	Five Replayer tools operating in coordination . . . . .	34
3.3	Replayer's <i>Meta Tool</i> , Used for component management. . . . .	34
3.4	A visual interface for selecting data . . . . .	35
3.5	Replayer's Time Series Component . . . . .	37
3.6	A remotely selected Time Series . . . . .	38
3.7	Replayer's event series component . . . . .	39
3.8	Replayer's histogram component . . . . .	40
3.9	Replayer's Graph Drawing Module . . . . .	41
3.10	Replayer's Correlation Component . . . . .	43
3.11	Replayer's Google Earth Bridge Component . . . . .	44
3.12	Replayer's Time slider component and quicktime videos . . . . .	46
3.13	Screenshot from DRS . . . . .	47
3.14	Positioning media on the master timeline. . . . .	51
3.15	DRS's Video Viewer . . . . .	52
3.16	DRS's <i>Audio Viewer</i> plays MP3 and WAV files. . . . .	53
3.17	DRS's Track Viewer Component . . . . .	54
3.18	DRS's Image Viewer . . . . .	54
3.19	DRS's Transcript Editor. . . . .	57
3.20	DRS's Annotation Table Viewer. . . . .	57
3.21	An example of a very simple coding scheme. . . . .	58
3.22	Illustration of different coding types. . . . .	59
3.23	DRS's Document Viewer. . . . .	62
3.24	DRS's Head and Hand Trackers. . . . .	63
3.25	DRS's concordance viewer. . . . .	66
4.1	Replayer's instrumentor running in visual studio . . . . .	73
4.2	DRS's LogFile WorkBench . . . . .	77
4.3	Multiple networked instances of Replayer . . . . .	82
4.4	Replayer's System Architecture . . . . .	86
4.5	A partially selected event series. . . . .	88
4.6	A partially selected histogram. . . . .	88
4.7	Creating a new server project from a standalone project. . . . .	98
4.8	Client update process to merge server changes. . . . .	99
4.9	Merging client changes with the server. . . . .	101
4.10	The QCCI synchronised digital clapperboard . . . . .	104
4.11	Interrelation of state and event selections . . . . .	108
4.12	File/analysis Synchronization using absolute times . . . . .	109
4.13	File/analysis Synchronization using absolute times . . . . .	110
5.1	The PDA interface to Treasure. . . . .	114

---

5.2	Using Replayer to show the context of a pickpocket . . . . .	119
5.3	Performing a mutual information analysis in Replayer. . . . .	124
5.4	An XML based log from Day of the figurines . . . . .	127
5.5	Seeing what players did at locations. . . . .	129
B.1	Ekmans 6 basic emotions . . . . .	189
B.2	Detecting facial expressions on a rollercoaster . . . . .	190

# Chapter 1

## Introduction

### 1.1 Introduction

Ubiquitous computing is increasingly a part of everyday life. The miniaturization of technology and the continuing fall in prices has allowed computing power to be embedded and leveraged in a wider range of situations than ever before. People performing everyday tasks are making use of ubiquitous computing technology often without even realising it. When they vote in the latest reality television phenomenon; when they call friends on their mobile phones to find each other in a crowd and when they change which route to take when travelling because their satellite navigation tells them that there is traffic ahead are just three simple examples. There are so many applications of ubiquitous computing pervading our everyday lives, that it becomes important to consider how such technology might affect the process of qualitative data research.

At one level there is a need to explore how technology can be leveraged to support *in situ* studies of the technology in use, but there is also a more general case. The pervasiveness of this technology is such that it has simply become part of most peoples' lives. A person in the UK without a mobile phone is rare and more recent smart phones such Apple's *iPhone* and phones running Google's *Android* operating system are starting to use global positioning systems (GPS) to deliver location aware services. Large numbers of cars are now shipped with satellite navigation; shops equip their stock with RFID tags that tell the shop's computers what is in and out of stock and display. Increasingly our everyday lives are affected by ubiquitous computing and thus *any* study of our everyday interactions must also be aware of how we interact around these technologies, how we fit them into our lives,



and how they affect the character of our interactions with other people and settings.

The notion of technomethodology was originally coined by Graham Button and Paul Dourish to refer to the making of the invisible characteristics of computer systems visible and accountable, i.e., available to inspection and analysis [38]. In its original context, technomethodology was intended to be a notion that would serve to elaborate the computer to end-users. It is appropriated here as a device for reasoning about user evaluation in ubiquitous computing. Specifically, it is exploited to promote the recommendation that we develop *technologies of observation* which make interaction in ubiquitous computing environments visible and accountable. What do we mean by technologies of observation? We mean things like microscopes, telescopes, particle accelerators, etc. We mean a technology that helps you see something. We take it that observation is the basic problem in user evaluation of ubiquitous computing environments, which move computing away from the desktop and distribute interaction across heterogeneous devices that exploit invisible sensing systems. Users are online and on the streets [25] they interact via different interaction mechanisms [30], and interaction is mediated by invisible sensing systems [24]. The *asymmetrical* and *fragmented* nature of ubiquitous computing [32], makes interaction difficult to observe let alone evaluate from the outset. Right now effort is largely focused on observational methods. Many of these methods are derived from the social sciences. They include quantitative [37], qualitative [30], experimental [52], and *in the wild* approaches [43], and they involve a range of elicitation techniques or procedures - e.g., interviews, experience sampling, diary studies, and ethnographic observation. These efforts are complemented by the development of frameworks for analysing the data gathered, which seek to provide a stable foundation for user evaluation in ubiquitous computing [189]. The problem with the current concern with method is that it does not address the basic problem of observation: if you cannot see interaction or see it adequately in the first place, then you cannot evaluate it or do so adequately either. Thus, the concern with method needs to be complemented by the development of observational technologies. Our premise is a straightforward one: in natural science a great many phenomena need to be made observable and this is done through the development of new technologies. The history of scientific development is the history of technological development whereby the objects of scientific inquiry are made visible and accountable [99]. The suggestion is that we must do likewise to support the study of ubiquitous computing systems.

This thesis will focus on methods of supporting qualitative researchers who are studying

peoples' interactions *with* ubiquitous technologies and with the understanding that the same techniques could be applied to studying peoples' interactions *around* the same technology, by constructing new qualitative data analysis software and demonstrating that the use of such software can add value to a study.

## 1.2 Approach Taken

We will begin by defining exactly what is meant by the term ubiquitous computing, and including some few examples of the types of technologies it applies to. We will look at how qualitative research methods (and in particular ethnography) are currently used in the study of ubiquitous computing. We will then clearly lay out the inherent challenges associated with applying qualitative research methods to study interaction with this type of technology. Following that we will describe the current state of the art in computer assisted qualitative data analysis software (CAQDAS) including some examples of how CAQDAS software is being used by qualitative social scientists 'in the wild.' We will describe the process by which qualitative methods can be applied to system log files in their most basic state. All this will lead to a set of design guidelines for a new generation of CAQDAS software capable of supporting social scientists in the study of peoples' interaction with and around ubiquitous computing systems.

Next we will describe two new software packages: *Replayer* and *Digital Replay System (DRS)* which approach those challenges from different directions. We will see that Replayer is largely concerned with capture, replay and more crucially representation, synchronization and filtration of system log data and its use in combination with other media types such as video and audio, and that DRS is aimed squarely at evolving more traditional CAQDAS software techniques (as described in the related work section) to better integrate with these new types of data. We will start by giving a comprehensive overview of the functionality of each system. Next we will drill down and focus on the design and implementation of three key features of both systems, namely:

- Distributed software architectures
- Log file handling
- Synchronization (both technical and methodological)

Next we will describe two case studies, one of the use of Replayer and one of the use of

DRS. We will show in these case studies how the coordinated use of system log data and other recorded data types such as video can help to build up a description of the character of an interaction, and thus support the process of qualitative description of those interactions.

Proceeding that we will compare the approaches of the two systems highlighting the specific strengths and weaknesses of each and describing how future work can solve the inherent weaknesses in each system by combining the two approaches into a unified whole. We will then revisit the original challenges laid out in the related work section associated with performing studies of people interacting with and around ubiquitous computing systems and see how the software has addressed those challenges, with specific reference to the case studies. We will also revisit the key design challenges defined in the related work section and show how both Replayer and DRS conform to those ideals. Finally we will sum up the academic contribution on this work and show in the conclusion that using heterogeneous media including system log data as qualitative resources really can support the process of qualitative data analysis.

## 1.3 Contributions

There follows a summary of the key academic contributions of this work. These contributions will be described in more detail in the final chapter of the thesis.

- Demonstrate the value that can be added to a qualitative analysis by combining log data with other media.
- Show how the coordination of system logs and other media can help us discover information in the data that would otherwise have been inaccessible.
- Show an ability to perform studies in locations/settings that would otherwise have been impossible, or at least very difficult
- Demonstrate how computer aided qualitative data analysis software has to evolve to support understanding interaction in a world increasingly affected by ubiquitous computing technology
- Demonstrate how creating accessible qualitative data analysis tools allows people to study particular settings or technologies who could not have studied them before.

- Show how we have defined, implemented and demonstrated an effective framework for storing and synchronizing log files.
- Show how this framework can be exploited to achieve coordinated views and selections of recorded data using techniques of brushing and linking.
- Show that with the use of a distributed software architecture some of the basic problems associated with viewing high dimensional data can be addressed.
- Show that we have developed two pieces of proof of concept software, Replayer and Digital Replay System with radically different approaches to supporting the use of heterogeneous media types as a resource for qualitative analysis of interaction with and around ubiquitous computing systems, and show how those disparate approaches can be combined to create the next generation of CAQDAS tools.

## 1.4 Thesis Statement

The current observational technologies available to social scientists are insufficient to support the qualitative analysis of peoples' use of ubiquitous computing systems. New software needs to be created that can help to reveal the character of this interaction by use of coordinated, synchronized views of heterogeneous media, including system logs. This can then be used as an accountable resource to add significant value to an evaluation by helping researchers find information in the data that would otherwise be inaccessible; giving researchers the ability to perform studies in locations and settings that would otherwise be impossible, or at least very difficult; and by creating accessible qualitative data analysis tools allows people to study particular settings or technologies who could not have studied them before. The development of these tools must be directed by working closely with social scientists studying people using ubiquitous computing systems.

## Chapter 2

# Related Work

### 2.1 Introduction

This chapter will begin with an exploration of the evolution of ubiquitous and mobile computing as an important area of computing science research. Following that we will describe the specific challenges associated with applying qualitative data analysis methods to ubiquitous computing systems. We will examine the state of the art for current computer assisted qualitative data analysis software, both commercially and academically produced and cite some examples of social scientists using such software 'in the wild.' We will then look at how qualitative social scientists may currently use system log data as a resource and produce some guidelines for the creation of new tools to support the exploitation of such data.

### 2.2 Ubiquitous Computing

#### 2.2.1 The Origins of Ubiquitous Computing

In 1991 Mark Weiser published a landmark article in *Scientific American* called *The Computer for the 21st Century*, containing a mission statement that became the foundation of ubiquitous computing research world wide. [225] The article is summed up most concisely with the following quote:

*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*

This is a direct evolution of an idea defined by Martin Heidegger in his seminal work: *Being and Time* [104]. Heidegger proposed the idea of tools being either ready-to-hand or present-at-hand. The premise is often described with the analogy of a hammer: when one uses a hammer, one focuses on the task of hammering in a nail, rather than the hammer itself, that is, the task should be the focus of one's attention rather than the tool. Weiser postulated a world of invisible computing, predicated on two core ideas:

- Technology should not require the user's active attention.
- Technology should be available for use at a glance.

Weiser's vision came on the back of proposals at Xerox Palo Alto Research Center (PARC) in 1987 by Bob Sprague and Richard Bruce to create wall-sized, flat panel displays from large-area amorphous silicon sheets. At the same time PARC's work practices and technology department, lead by Lucy Suchman was looking at the way computers were embedded into the complex social framework of everyday life [204]. By the time PARC started to produce examples of this new ubiquitous computing, such as *LiveBoard* [74], *ParcTab* [217] and *Active Badges* [218], PARC had started to focus less on the practice of human-computer interaction and more on human-human interaction, relegating technology to the place in the background that Weiser's vision said was where they should be.

Sadly, with his early death in 1999, Weiser never lived to see just how influential his research would become. In 2000 The European Commission published an IST call for research proposals in the area of future and emerging technologies named *the disappearing computer* [226]. With the proliferation of mobile phone technology the public began to become aware of the impacts of ubiquitous computing, even if the term may remain unfamiliar outside of the field. The CIA World Factbook [1] reported in 2008 that there were 69.657 million mobile phones in the UK alone, which with a population of 60.776 million reveals the staggering fact that for the first time there were more handsets in the country than people. With such a large percentage of the population carrying potentially significant computing power with them at all times, and the demystification of the technology, the importance of research into ubiquitous computing has never been higher.

In 1998 Don Norman published a book called *The Invisible Computer* [164] in which he approaches the idea from a background of design and psychology. He describes the traditional personal computer as complex, difficult to use and expensive to maintain. He directly reflects Wieser's adaptation of Heidegger's ideas when he asserts the need to:

*Design the tool to fit the task so well that the tool becomes part of the task, feeling like a natural extension of the work, a natural extension of the person.* [164]

This same idea is discussed extensively in Paul Dourish's 2001 book *Where the Action is* [67] which describes the principle as 'embodied interaction', aiming to show how an understanding of this principle can affect the design process of modern software and hardware.

With the emergence of ubiquitous and mobile computing as an important and accepted area of computing science research the number of conferences and journals has steadily increased. Conferences such as Ubiquitous Computing and MobileHCI, focus specifically on this area, though the research tends to stay under the wider banner of Human-Computer Interaction, with conferences like CHI and CSCW still being major goals for the publication of ubiquitous computing research.

### 2.2.2 What do we really mean by a Ubicomp System?

Just a few years ago, when discussing ubiquitous computing *systems* authors were typically referring to academically developed proof of concept systems, such as those developed by the *Equator* Interdisciplinary Research Council. These systems frequently exploited the most up to date available technologies for mobility (portable digital assistants, mobile phones) connectivity (802.11 Wifi, general packet radio service (GPRS)), Positioning (Global Positioning System (GPS), Ultrasonic Positioning), And all manner of additional sensors such as accelerometers, heart rate monitors and galvanic skin response monitors. Many systems were developed exploring many different facets of human computer interaction from many different angles: artistic, playful, educational etc. Some notable examples include:

#### **Tourism**

The City project began by performing an ethnography on city visitors [31], then began to focus on designing systems to support collaborative interaction as part of the process of tourism, beginning with the *mack room*, set in the lighthouse museum [30] wherein users took part in a shared museum visiting experience with only one of the three users physically present, the other two being in a virtual reality environment and a web page environment respectively. City continued by scaling the experiment up to the idea of general tourism, culminating in the *George Square* project [20]. Other examples from outside of equator include the application of robotic avatars to museum visiting [183], systems to make guides more context aware [16] and much focus on education and young person engagement, e.g.

[110, 213]. Indeed, so many interactive tourism systems have been developed recently as to prompt the introduction of a new conference on the subject (HCITTOCH: Human-computer interaction, tourism and cultural heritage), and a comprehensive review of the state of the art in digital tourism [124]

### Mobile Gaming

Examples include *Can You See Me Now?* [25] and *Uncle Roy All Around You* [78] both of which were nominated for BAFTA awards. *Can you see me now?* Was a chase game based on the idea of internet players chasing real players through the streets, that is, the internet players were chasing a digital representation of the street players through a digital representation of the city, but the street players were actually running in the real world, being chased by digital *ghosts*. The system explored some of the challenges of location based technology, demonstrating the strategies employed by players to use and exploit the uncertainties in GPS and WiFi. It also contributed towards the development of the Equip platform [98] designed to support the creation of experiences using many different physical technologies. *Uncle Roy All Around You* was another game which featured players in the street, this time trying to track down the elusive and eponymous Uncle Roy. The game explored the use of self reported positioning as a technique for location awareness, demonstrating by means of both implicit and explicit positioning that humans interpret position in subtle ways beyond just absolute location [23]. Outside of equator, ubiquitous computing-based games have flourished as a research topic, or at least as a means to explore other research topics. Some examples include: *Pirates!* [28] which combined physical, location-based gameplay with virtual, screen-based gameplay - in demonstrations of the game, players explored the same physical environment while simultaneously navigating a fantasy archipelago depicted on their PDA screens; *The Drop* [200], a theoretical game, never actually produced, but proposed by Smith and La Marca at Intel Research as an example use of its Place Lab [139] multi-platform indoor location technology; and *The Invisible Train* [215] which uses augmented reality to place a digital toy train on the same track as a physical one. Of course these examples can only begin to scratch the surface of ubiquitous computing gaming applications. Several labs, for example the Exertion Games Lab at RMIT University, Melbourne<sup>1</sup> or the PLAY research studio at the Swedish Interactive Institute<sup>2</sup> focus exclusively in this area, indeed the conference *Advances in Computing Entertainment (ACE)* and its sister

---

<sup>1</sup><http://exertiongameslab.org>

<sup>2</sup><http://www.tii.se/>



journal *Computers in Entertainment (CIE)* frequently publish articles discussing ubicomp games and the range is ever expanding.

### Education

*The Hunting of the Snark* [101] was a project designed to help children explore the ideas of cause and effect. It took the form of an adventure game using a variety of tangible interfaces. *The Ambient Wood* [220] and its indoor component *Classroom Ecology* were designed to help children explore the world of ecologies. Set in a local wood, the system used location awareness technology to deliver information about nearby living organisms to a mobile device carried by the children giving them access to information about the invisible aspects of the wood around them. The children used a probe tool to measure information like moisture and light leading up to a classroom discussion about the local ecology. Again the range of applications of ubiquitous computing technology is extensive. Some examples from outside equator include *Virtual Classrooms* [143] which looks to replace traditional video based e-learning systems with the eponymous virtual classroom (essentially a collaborative virtual reality environment) in which each participant has an avatar; *Mobile Stories* [76] which looks at the collaborative construction of stories on childrens' mobile devices; and much work has focussed on using ubicomp to specifically support children with autism, e.g. [84, 111, 208]. Poole et. al elaborate in [173] exactly how one might go about deploying ubicomp technologies in schools - with a particular focus on health interventions.

To create an exhaustive list of examples of ubiquitous computing systems is beyond the scope of this work. The field is sufficiently large to merit several annual international conferences including *Ubiquitous Computing*, *Pervasive Computing* and *Mobile Human Computer Interaction* amongst others. An important point to note however is that ubiquitous computing is no longer just something done by researchers. Ubiquitous computing hardware is fast becoming considered *commodity* hardware. A person without a mobile phone now seems like an aberration. Those phones often include location aware services supported by cell positioning and even GPS. They also often contain Bluetooth or WiFi allowing users to wirelessly link together multiple devices to, for example, use a hands free kit or share files between two peoples' phones. As the technology continues to drop in price, and the services become more widely known, this is a trend that is likely to continue.

## 2.3 Evaluation

Several articles have remarked on the difficulties and challenges facing ubiquitous computing evaluation [5, 58, 109]. Despite this, the number of systems that have gone through truly iterative design cycles is quite small (e.g. [2, 121, 157, 161]) as compared with more traditional desktop based systems which frequently undergo more familiar types of lab based usability evaluation. This may well be a consequence of how difficult ubiquitous computing systems can be to evaluate. We begin by discussing examples of iterative design, followed by a discussion of some places that existing formative and summative techniques have been applied to greater or lesser success.

One of the first ubiquitous computing systems to receive extensive study from a user perspective was eClass (formerly Classroom 2000) [2, 4]. EClass was a sensor-embedded environment in which, at various times, lecture slides, written comments, video and audio, and other classroom activities were captured and organised and students could later access this information on the internet. EClass was used and evolved over the course of more than three years and during this period, various qualitative and quantitative evaluations were completed, leading to the system as deployed evolving and changing. As a result of this work, and other related projects, Abowd et al. developed the concept of a living laboratory, an environment occupied by real users in which a research system is deployed, evaluated, and updated as it is used over time [3].

This was one of the first successful long-term deployments of a ubiquitous computing application, and certainly one of the first to include regular study of and feedback from end users. While eClass was an excellent proof of the potential of Ubicomp applications, and a fine example of iterative design, neither the evaluation techniques nor the prototyping tools used in the project particularly lent themselves to rapid iteration. Another early system, Tivoli [157], was developed to support meeting activities. Moran et al.'s experiences with Tivoli led them to develop tailorable tools so that they could better adapt the application to varying user needs. The applications just described provide solid examples of iterative design in ubiquitous computing, and involved mainly summative evaluations. In contrast, Jiang et al. and Mynatt et al. went through multiple iterations in the design of prototypes of new ubiquitous computing applications. Jiang et al. developed and compared two prototypes of large-screen displays to support fire fighters' incident command centres, and then developed a third, improved display based on the results of evaluation [121]. That evaluation principally

involved showing the prototypes to fire fighters and asking for feedback. Mynatt et al. developed multiple designs of a digital family portrait, all before a working prototype was completed [161]. Their iterations involved a combination of techniques such as surveys, interviews, and Wizard-of-Oz evaluation [161]. In addition to these examples of iterative design, numerous developers have successfully used either existing formative evaluation or existing summative evaluation in the design of Ubicomp applications.

Formative techniques, used for requirements gathering and understanding problem spaces, at the stage before actual systems are built, are probably among the most common found in Ubicomp application development at present. Areas of study include the home [54, 197], information management [22], hospitals [13], and various other settings. One early piece of work in early-stage evaluation of a non-working prototype was Oviatt's use of Wizard-of-Oz evaluation to test a multi-modal map application which combined speech and pen input in different ways [169] (though Dahlback was first to highlight the uses of Wizard-of-Oz systems in this domain [63].) Wizard-of-Oz evaluation has subsequently been applied to other Ubicomp systems, e.g. [161]. Finally, Jianget al. used a form of expert user evaluation to study non-working prototype systems [121].

Summative evaluation as an often used alternative for studying ubiquitous computing systems. Oviatt, for example, made use of a wearable computer to test the robustness of speech in unconstrained settings [170]. When combined with pen input, the system was shown to be remarkably effective in recognizing speech. A related study conducted by McGee et al. compared maps and post-it notes with a tangible multi-modal system [151] using a controlled, lab-based study. Consolvo et al. used Lag Sequential Analysis in a summative evaluation to gather quantitative data about how their system was used [51]. Although an interesting and informative technique, Lag Sequential Analysis requires hours of video to be coded by hand into different categories. Researchers investigating alerting displays provide one example group where extensive evaluations have been conducted (e.g. [49, 211]). The alerting display community has put significant effort into developing a deep understanding of interruptibility [113]; parallel and pre-attentive processing [68, 108] and the attentional impact of different types of animations or other alerts [15, 60, 116, 147]. However, these studies normally involve extensive lab-based testing rather than discount, real-world evaluations of actual systems.

When evaluating the user interfaces of ubiquitous computing applications, laboratory studies can help to discover some interface and navigation problems [26, 123], especially

when the study is designed to reflect some aspects of the setting in which the real use will take place [131]. Even simple heuristic evaluation [172, 198] can be useful, as can paper prototyping, as Carter and Mankoff assert [43]. However, as Zhang et al. and Monrad et al. point out, situated experimentation is vitally important for understanding the unexpected behaviour. [162, 234]. Tools do exist to support the evaluation of ubiquitous computing systems, one such technique is experience sampling [115, 141]. Intille et al.'s [117] aimed to make the experience sampling device context aware, asking questions of the user only when particular conditions are met by its sensors. Momento, developed by Carter et al. aims to support the logistics of fieldwork by supporting the capture of various qualitative materials including system logs and experience samples [44]. MyExperience, by Froehlich et al. also supports this process by capturing system logs from mobile phone use. [86]. In [168], Oulasvirta et al. focus on hardware configurations for deploying effective mobile usability labs so called "highly instrumented interaction". These build on work on previous wearable camera systems such as [145, 179, 191]. Indeed Oulasvirta goes on, along with Raento and Eagle, in [176] to argue that smartphones are becoming an increasingly powerful tool for social science - citing their programmability, ubiquity and cost effectiveness as a method of capturing information about users' behaviour.

The tools and methods described in this thesis will focus on supporting qualitative approaches to the evaluation of ubiquitous computing. In the environment in which the tools discussed later in the work were developed, the local social science expertise was largely focussed on the practice of ethnography, so it is this method of qualitative analysis that has largely shaped the development process; in particular because a participatory design approach was used (more deliberately in the development of the Digital Replay System) to explore the requirements for tools to support the analysis within and around these new interaction media.

With its focus on the situated nature of interaction and the social character this reveals [196] ethnography, or more properly, ethnomethodologically-informed ethnography, is especially relevant to understanding the purchase of ubiquitous computing in the wide array of social settings that are intrinsically part of the definition of ubicomp. It is appropriate now to briefly examine what is meant by the term ethnography in the context of this thesis.

The term ethnography is really a collective term for heterogeneous practices and ways of thinking. To say that ethnography is concerned with understanding situated action is not strictly true, rather that is the concern of ethnographers working in the ethnomethodological

tradition [89]. Ethnomethodolgy does not consider ethnography to be a specialised way of examining action, but rather something we all do as part of our everyday lives [88]. When we ask someone to describe some action that took place, such as a child's day at school or describing a night out to our friends we are performing an ethnography. So if ethnographies are such a mundane part of our lives how can it be considered to be an effective approach to social science research, and in particular a tool for interaction evaluation, to affect systems design and understanding in computing science? The distinction is clearly in the way the ethnography is analysed. In a professional ethnography the ethnography is described in terms of a formal analytic schema for intense analysis, and packaged in a report, while the lay ethnography is simply casually understood. [88].

There are two distinct methods by which ethnographies can be reported, first through formal analysis which is concerned with describing the materials in the form of coding schemes, taxonomies, narratives, models and other situationally absent descriptions [90]. Second we have the ethnomethodological approach which describes the material in *thick description* [184] of the practical action and practical reasoning exhibited by subjects as their shared activity progresses [196]. Both these techniques share an understanding that the process of description is reflexively connected to an analysis [185], but each approach is concerned with this reflexivity in different ways.

Much of the process of formal analysis is concerned with analytic reflexivity, in which critical self-reflection is used to understand the ways in which the act of ethnography shapes our understanding of a setting and the actions within, and seeking solutions to the professional belief that that cultural and subjective biases are intrinsically built in to any analysis [146]. Ethnomethodology however, despite accepting that some awareness of this subjective bias is relevant, tends to reject analytic reflexivity [27], concerning itself with the reflexivity of accounts [88].

When considering the reflexivity of accounts, instead of focusing on the effect of the analysis on the study, and formal representations, an ethnomethodologically-informed ethnography focuses on the everyday settings that people inhabit specifically that which is observable and reportable, the so-called *work practices* that shape those settings and the interaction that occurs within them [62,89]. Such studies have played a vital role in computing science research since Lucy Suchman's 1987 seminal work in the field of human-computer communication, *Plans and Situated Action*. [204]. So called *ethnographies* have become a staple feature of computing science research and systems design, to the extent that a corpus of

such studies populates much HCI, CSCW and DIS literature, elaborating the character of interaction and collaboration, of organisation and of technology and its use in the home and workplace. This thesis will demonstrate ways of supporting such studies into the more difficult to capture field of ubiquitous computing, and specifically *in situ*.. Some early experiences of using ethnography to explore the use of ubiquitous computing systems include Brown et al. [30] [32] and Woodruff et al. [231]

### 2.3.1 Specific Challenges of Evaluating Ubiquitous Computing Systems

The study of ubiquitous and mobile computing systems presents a series of new challenges previously unseen in the interaction analysis of more traditional desktop based systems. Ubiquitous computing systems are intrinsically linked to the setting in which interaction occurs, and how this setting may be explored to shape a user experience. Features like context-awareness allow the technologies to dynamically adapt to different places, users, activities and situations. As these kinds of technology mature and become increasingly pervasive it is important to complement their design with a thorough understanding of their impact on situated action in order to ensure that the technology fits with the social circumstances of their use.

The study of such technologies *in situ* necessitates a shift from the traditional paradigms of analysis. Observation (at least in the traditional sense) alone may not be sufficient to discover the whole story. The traditional methods used to capture information about a system such as video and audio recordings, photographs and field notes are simply unable to adequately describe the context in which an interaction occurs. The reason for this is intrinsically tied up with the definitions of mobile and ubiquitous computing. Analysts have to cope with such rarely previously encountered challenges as the following:

- *Mobility*. Users of ubiquitous systems are often mobile. They move across extended physical areas, quickly at times, sometimes even running, which can make the documentation of action and capturing of video material difficult at best.
- *Small Displays*. Interaction frequently involves the use of small displays such as handheld computers and mobile phones. This makes it difficult to see users' interactions with the system.
- *Headphones*. Users often have audio information provided through headphones which

becomes unavailable with traditional capture techniques like video recording. This means that the analyst may miss much key information.

- *Occlusion.* Closely related to the previous two issues, the situation of an interaction means that the environment, or even the users themselves will frequently occlude information that might otherwise be captured by video cameras or simple observation. The same is true of audio cues, environmental noise can easily occlude key audio information.
- *Non-collocation.* When interacting with collaborative ubiquitous systems users are frequently interacting with other users who are not collocated with the user. Again this affects the analysts' ability to capture all the required information about the context of the interaction, unless multiple analysts and capture devices are involved, and even then this presents the challenge of the synchronisation of captured data.
- *Invisible interaction.* Users often interact with invisible sensor systems such as Global Positioning Systems or video tracking, which can make it challenging to understand why users are acting in a given way and how the sensing systems are actually behaving.
- *Distribution of Interaction.* Interaction may be distributed across different applications and devices. Interaction is thus not only located in different physical locations but may also be mediated through different applications and devices, which makes it difficult to develop a coherent description of interaction.
- *Interaction Time.* Certain communication channels such as mobile phone text messages (SMS) or email are not continuous, that is, a message may come in at one time, but a user may examine it much later then wait till a convenient time to respond. This complicates the process of conversational analysis as the context of the times when a user receives a message and when they act on it may be very different.
- *Technological breakdowns.* Because ubiquitous computing systems frequently rely on the use of diverse technologies all being used together, we often assume that these technologies will smoothly interconnect and interact. In reality however, these technologies do not always work perfectly together, potentially creating confusion for both the user and the analyst in the interaction process.

As can be seen from these examples, describing the interactional character of a ubiquitous computing system is different to describing that of more traditional kinds of computing

environment. The problem becomes one of reconciling many fragments of recorded interaction into one cohesive description framework. Additional data has to be recorded and supplied to the analyst as part of their resources. The best source for this data is system logs. There is of course nothing new in exploiting system logs to support the understanding of interaction. HCI researchers have been doing this for decades and techniques for doing so are documented in most good HCI textbooks. However, ubiquitous computing goes beyond logging machine states and events to record elements of social interaction and collaboration conducted and achieved through the use of ubiquitous applications as well. For example, audio, location information and textual interaction may be logged alongside machine states and simple UI events. System recordings make a range of digital media used in and affecting interaction available as resources for an analyst to exploit allowing an understanding of the distinctive elements of ubiquitous computing and their impact on interaction. The challenge then is one of combining a burgeoning array of internal resources with the more traditional external resources gathered by an analyst to support a thick description of the character of interaction in complex digital ubiquitous computing environments. Qualitative researchers, and in particular, ethnographers will already be familiar with the process of *bricolage* [64] in which multiple disparate resources are combined to understand social behaviour. It will be in part the work of this thesis to demonstrate how one may support the bricolage process in including these more unfamiliar resources - and in making them into accountable objects.

## 2.4 State of the art in CAQDAS

Here we will first explore the current state of the art for the analysis of ubiquitous computing systems. we will look at some currently available examples of computer aided qualitative data analysis software (CAQDAS) in both commercially and academically developed toolkits, and provide some concrete examples of qualitative social scientists actually using them to support their research. We will then describe an example of researchers attempting to exploit system log data *by hand*, that is, without additional tools to turn the logs into understandable and accountable objects, in order to demonstrate to complexity of that task. This will lead us to a set of key design guidelines for the next generation of CAQDAS tools which the proceeding chapters will then discuss.



### 2.4.1 Replay

There are a large number of tools available which provide some form of replay, either to support transcription and coding of video and audio data, or to support the wider analysis process. In this section we will examine many of them, starting with commercial packages, then looking at academic work surrounding replay.

### 2.4.2 Commercial Tools

#### Transana

Initially developed at the University of Wisconsin, *Transana* <sup>3</sup> is perhaps the most widely used CAQDAS tool across the social sciences. It is a transcription program allowing users to transcribe both video and audio and mark areas as interesting. It uses audio waveforms to support the transcription process. In 2007 Transana made a paradigm shift from freely downloadable tool to paid software, somewhat reducing its breadth of uptake. Transana is still so popular however, that as part of the design process for Digital Replay System a decision was taken to use the same keyboard commands to control the system, and indeed to support importing and exporting to Transana's output format.

#### Atlas.ti

*Atlas.ti* <sup>4</sup> is probably the most versatile commercial tool on the market at this time. Its primary focus is coding, but unlike the other coding tools discussed below, It supports the coding of text-based data as well as video and audio. It provides a basic automation of coding based on regular expression searching within text, as well as providing integration with Microsoft Office <sup>5</sup> and SPSS <sup>6</sup>. It supports a project-based approach to corpus management, allowing a user to keep all files organised into projects. Each item (referred to within Atlas.ti as a unit) such as a code, or a segment of text is annotatable. It also provides support for techniques such as mind mapping.

---

<sup>3</sup><http://www.transana.org>

<sup>4</sup><http://www.atlasti.com>

<sup>5</sup><http://office.microsoft.com>

<sup>6</sup><http://www.spss.com>

### The Observer

Developed by Noldus, *The Observer* <sup>7</sup> was designed originally for studying animal behaviour patterns and has been adopted as a more general coding solution within the social sciences, particularly in behavioural psychology, but boasts case studies of its use in neuroscience, psychology and zoology. A module based system, users can buy modules to perform different parts of their analysis, including the base module which supports coding of video data and visualisation of those codes for analysis. Additional modules offer support for multiple videos, code comparisons, confusion matrices, screen capture and a mobile module designed to run on handheld computers. Like Atlas.ti, the observer is Microsoft Windows based, and extremely expensive.

### INTERACT

Mangold International's *INTERACT* <sup>8</sup> is another windows based observational analysis solution that supports the process of coding videos, then provides some simple visualisation tools to support analysis of the coded data. It is unique amongst the commercial tools in that it provides support for extensibility in the form of the Interact Extension Language: an, albeit limited, scripting language which allows the user to perform transforms on the data, as well as supporting import and export to specific formats. Basic export for tools like SPSS and Excel is provided. INTERACT's marketing department makes the rather bold claim of being able to analyse your data automatically, though on closer inspection this actually refers to its support for basic statistics and visualisations. A key factor here, which will be discussed later, is that these programs cannot and should not be expected to 'analyse your data automatically'. That is the task of the social scientists. The programs should be there simply to support that task.

### StudioCode

Unlike all the other software described here (with the exception of Transana which is cross-platform) *StudioCode* <sup>9</sup> is designed to run on Apple's OSX platform. It provides a solution for capturing data straight from firewire (IEEE1394) video cameras and a suite of tools for coding and transcribing those videos. StudioCode has a deceptively clean interface hiding as it does a very versatile coding system supporting multiple coding schemes (templates),

<sup>7</sup><http://www.noldus.com/site/doc200401012>

<sup>8</sup><http://www.mangold-international.com>

<sup>9</sup><http://www.studiocodegroup.com>

live coding, lead and lag times and multiple coders. It is probably the most polished of the software described here.

### 2.4.3 Academically Developed Tools

As can be seen from the selection above, there is no shortage of commercial video analysis tools available on the market, however there are also a number of academically developed tools performing similar functions. This section will examine each of these, as a contrast to what is available commercially.

#### I-Observe

An early system synchronising video with log data was developed by Badre et al. [11] which used a video tape based system and made use of captured event streams to synchronise time-stamped events with the time-code on a video. This was designed for static, lab based usability testing and was somewhat limited by the technology available at the time, however this system could be considered to be well ahead of its time in the way it combined heterogeneous data types for analysis, and is perhaps the most closely equivalent to the core themes of this thesis, though its intended use may have been very different from those discussed here.

#### ANVIL

Developed in 2001 by Michael Kipp at the University of the Saarland, *ANVIL* was designed as a video annotation tool specifically for the purpose of analysing multimodal corpora [130]. It is a java based tool supporting the annotation of a single video stream and based on an underlying XML schema, with a simple GUI. It supports multiple tracks of annotation of video and the export of those annotations to common formats for further analysis in tools like Excel and SPSS.

#### Diver

The *Diver Project*, developed at Stanford University is another tool to support video annotation [171]. Designed to work with a single video, it nevertheless has a unique feature, that of the eponymous *dives* where users can manipulate the viewpoint of a video using a virtual camera viewfinder, allowing zooming, panning and rotation of the original video data. These dives can then be individually annotated and shared on the WebDive site for

collaboration with other analysts. Dive offers an interesting approach to repurposing the basic video data into a number of different perspectives depending on the purpose of a given analysis.

### **VACA**

Also developed at Stanford University *VACA* provides a toolkit for annotating or coding several simultaneous videos on a timeline representation [36]. A study was carried out comparing the use of *VACA* against a combination of Windows Media Player and Excel, showing that *VACA* allowed its users to perform a coding task nearly twice as fast.

### **ELAN**

Now in its third version *ELAN* was developed at the Max Planck Institute for Psycholinguistics [33]. It is a fairly comprehensive tool for the annotation of video data, primarily in the field of linguistic research. It supports annotation in tiers, what other projects might call tracks, so several simultaneous annotations can be applied to a single piece of media. It also offers a waveform panel similar to that of Transana to support the process of synchronised transcription. *ELAN* offers the facility to search within one's annotations, across multiple tiers, and allowing the user to jump to the location in the video of search results. Additionally *ELAN* supports the practice of collaboration by providing a network interface allowing two users of *ELAN* to collaborate on the annotation of a single piece of media.

### **NITE XML Toolkit**

The *NITE XML Toolkit*, developed at the university of Edinburgh [41] is primarily a set of open source libraries to support the analysis of heavily annotated corpora. It provides a GUI for the annotation of multimodal corpora supporting the annotation of textual, audio and video data. It includes a comprehensive querying language and some command line-based textual analysis tools. One of its goals is to allow better sharing of qualitative analysis data, though the Qualitative data exchange project [53] may prove to be a more comprehensive approach aiming to achieve the same goal.

### **Mixed Media Grid**

Developed at the University of Bristol, based on the same underlying code as Digital Replay System's immediate predecessor: ReplayTool [85], from which it was a development branch,

the *Mixed Media Grid* or MiMeG project is designed to support multi-site simultaneous collaborative video analysis. [82] It features video streaming and cross-site synchronisation, as well as some basic security precautions and allows its users to annotate video by drawing directly on top of a given frame.

### **Tatiana**

Post dating both Replayer and DRS, *Tatiana* is nevertheless one of only two other current video analysis tools that support the viewing of system log data, in this case referred to as trace data. [69] *Tatiana* allows synchronous playback of system logs in a given format with videos, supporting annotation. Initial tests performing a socio-cognitive analysis of a collaborative writing task have supported the assertion of this thesis, that including trace data can help to provide a more holistic view of an activity being analysed.

### **Chronoviz**

The newest software described here (from 2011), Chronoviz [80], developed at the University of California, is in practice perhaps the closest to the proof of concept software described in this thesis. Chronoviz allows a user to combine multiple streams of video with some types of logs (notably location traces and time series). Chronoviz also explores the use of interactive paper to take notes and control playback of the data streams.

## **2.4.4 Project Specific Replay Tools**

A large number of projects employ replay tools created specifically for analysing the logged data recorded during the execution of one program. Indeed, Replayer started its life as one such tool. It is of course the general applicability of tools like Replayer and Digital Replay System that differentiate them. However some examples of these include tools developed for analysing the following systems: *Savannah* [40] *ABSTRACT* [94], *CatchBob!* [165] and The Mack Room [30]. There are of course many more of these, but comprehensively listing them here does not really serve to further the themes of this thesis, rather it is enough to be aware that the practice of creating such tools does exist.

## **2.4.5 Capture and Replay of System Log Data**

In this section we will cover some tools created for the capture and display of system log data, of course the goal of this thesis is to cover the benefits of tools which use a synthesis

of both system log data and more familiar qualitative data types such as audio and video, but the practice of recoding and visualising log data has a long tradition within computing science and should be acknowledged here.

*GRUMPS*, developed at the University of Glasgow is a tool designed to capture java program usage data, by instrumenting Java Bytecode. [152] *GRUMPS* allows a user to dynamically choose the granularity of recording, then runs transparently while a program is executed recording for later analysis data about the execution of the program. Hilbert and Redmiles [106] developed an agent based system for capturing user interface events in java programs. Like *GRUMPS*, it performs its capture behind the scenes and was designed primarily for analysing web interfaces, to allow analysts to compare the actual results of an interaction with the expected results. *KALDI* [8] is designed to similarly capture Java usage data, though it also captures screenshots, allowing synchronised playback of the log data with the screenshots. *GUITESTER* by Okada and Ashi [166] uses a similar approach to that of *KALDI*. The event recorder *ObSys* [92] goes slightly further, working at an operating system level to capture every message from input devices that Windows puts in its event queues. Ivory and Hearst [118] provide a comprehensive view of the state of the art of usage data capture, making a comparative analysis of many of the systems that were available at the time: something which is beyond the scope of this section, thus reference to that paper is worthwhile for more information in this area. All the tools so far discussed have been designed to focus on usability testing of desktop (primarily web) applications. A more recent system, discussed above called *MyExperience* by Froehlich et al focuses on capturing trace data from mobile phones. [86]. It is the data from generalised capture systems such as this which will allow systems like Replayer and Digital Replay System to thrive as tools for the qualitative analysis of mobile and ubiquitous computing systems.

## 2.5 Example real uses of CAQDAS

Kuckartz [138], presents an exploration of the use of the MAXQDA <sup>10</sup> in particular highlighting the difference between case-oriented and cross-case visualisations of coded text. Kuckartz demonstrates the benefits of having the visualisations of codes directly related back to the source text - something that Replayer and Digital Replay System are deigned to do on a more multi-modal scale.

---

<sup>10</sup>[www.maxqda.com](http://www.maxqda.com)

Le-Roux et al. [142] discuss the use of CAQDAS software in an industrial context, citing it as an effective technique for analysis of target markets - in particular focussing on the effects of political and cultural transformation of those markets. They describe a process by which introductory courses to CAQDAS software have been proposed to company sociologists suggesting that such software is being used to support analysis outside of academia.

Murray [160] Describes a case study where NVivo <sup>11</sup> was used in a secondary analysis to try and prevent some of the pitfalls of the primary analysis - in particular a practice she refers to as code-fetishism, whereby the primary analyst used a coding scheme so diverse as to be virtually useless. By limiting the number of potential nodes in the secondary analysis, and making use of a hierarchical tree structure, available within NVivo instead of a completely flat *free* system, the data became more easily analysable, and more suitable to be interrogated in future analyses.

Di Gregorio and Davidson [65] Use case studies of the use of Atlas.ti and NVivo, to discuss the process of research design, based on an understanding of CAQDAS software. They highlight the key fact that a research project has to be designed with a knowledge of the way the software works in advance, in order to allow the user to achieve the goals they hope for by using said software. They discuss a need for explicit clarification of the design of a project beyond the typically implicit design that qualitative studies more traditionally use.

Varela et al. [212] discuss their use of Transana in a project studying a classroom workshop with their families. They demonstrate how they were able to make use of Transana to support the ethnographic process from project organisation through transcription and right through to analysis. The project focussed specifically on the way parents support their children in narratives using information related to their everyday lives.

Hesse-Biber and Crofts [105] meanwhile take a wider approach, exploring from a social scientist's perspective the availability of some of the different CAQDAS software discussed above, and examining how they fit in with different research styles, what benefits they offer and more practical considerations like system requirements and cost. They also consider the case of users' preconceptions of the capabilities of these programs - specifically whether the marketing hyperbole around particular programs is baffling its users into expecting more than the program may be able to give. In the commercial tools section (above) we see an example of this where INTERACT claims to be able to automatically analyse your data.

---

<sup>11</sup><http://www.qsrinternational.com>

There are of course many examples of social scientists writing about their experiences of CAQDAS software, this has been simply a small sample, however it suggests that there is an uptake of this kind of software within at least some of the social science community, and further, that community is willing to try out new tools and provide feedback about the way those tools are affecting their work practices and potentially benefiting their research.

## 2.6 Analysing Log files by hand

In [55] The authors discuss the practice (from an ethnographer’s point of view) of unpacking the data found in system logs to make use of that as a resource in an analysis of the mobile game *Uncle Roy all Around You* [78]. The authors assert that ethnographers frequently draw on a multiple resources to support the task of observing and analysing social life from *within* exploiting biographical resources [229], visual resources [175], technological resources [103], and a wide variety of others. They point out that ethnography is done not only through the immersion of a researcher in a setting but through the use of material resources as well. System logs, or *text logs* as they are described in [55] represent just another exploitable resource. However, the complexity of retrieving the value of that resource is a factor in the usability of these logs in their raw state.

To be usable, the log must first be *cleaned*, that is, reduced to contain just salient information. However, the definition of salient information is necessarily subjective. The authors freely admit that in this case the log cleaning is directed towards extracting conversational threads that are relevant to particular aspects of the study. In effect this means that for the resource to be applicable to more than one study, or more than one specific area of a study, it may have to be cleaned additional times, each time with a significant man-hour cost associated with the task. The next thing they note is that the log by itself is not terribly useful. It is only when combined and synchronized with the recorded media files, in this case with the transcriptions of audio clips where the synchronization is done at a textual level, that it becomes a useful resource for description. We can thus already see that it is the *combination* and *coordination* of system log data with other heterogeneous media types that creates an exploitable resource for qualitative analysis.

One very important point to be drawn from this can be best summed up by this quotation from [55]:

*Whatever the added extras it is hopefully clear that text logs do not, in and*



*as of themselves, contain data. Rather, the data must be produced through the analytically oriented working of resources internal to situated action and their combination with resources external to the setting of action. Data is constructed then and produced through the work-practices of the analyst.*

## 2.7 Design guidelines for ubiquitous computing analysis tools

Ethnomethodology, or ethnomethodologically-informed ethnography, can certainly be considered to be an effective tool for the analysis of interaction. However, predicated as it is on observation, there are, as we have seen, certain areas in which practitioners require some support. Specifically they require the development of new technologies of observation that make the invisible parts of these new interaction phenomena visible. It is the work of this thesis to demonstrate how such tools might be developed and used to support ethnographic, and wider qualitative approaches to analysing peoples' interaction, communication, and collaboration with, around and through ubiquitous computing systems.

In order for a CAQDAS tool to succeed as an effective tool for analysing ubiquitous computing systems, at least assuming we accept the challenges highlighted above, we must design systems that support the following key processes:

- Tools that allow viewing of system log data synchronized with other types of media
- Tools that enable researchers to extract multiple sources of information from recorded logs, and which allow them to edit extracted information and combine it with media from external sources to produce unique datasets.
- A replay system that exploits time stamps to coordinate the use of the multiple media in a dataset, which enables cross referencing and indexing to support the splicing together of multiple media, and which enables multiple media to be played side-by-side.
- Tools that support the production of representations from datasets and which preserve the relationship between representations and the media from which they are derived, and which enable source media to be recovered and viewed.

Figure 2.1 shows a flow-type diagram of how the record and reuse process could fit into both the fieldwork practices and the post-hoc analytical practices of a qualitative researcher

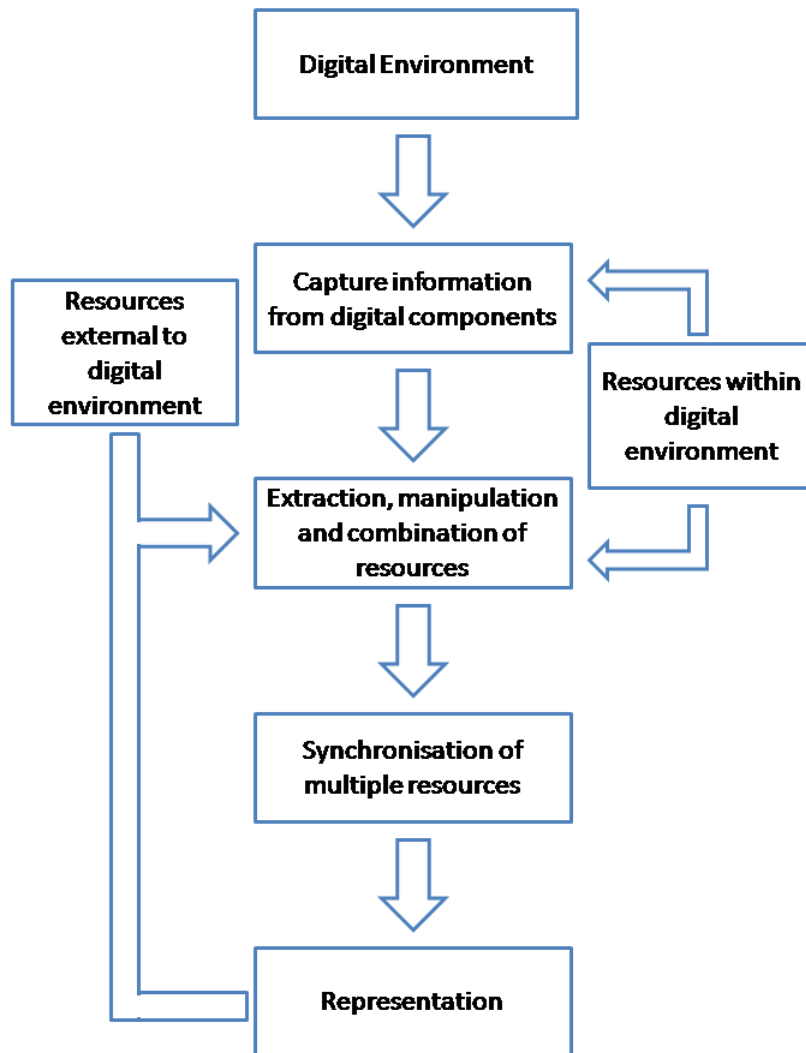


Figure 2.1: The Record and Reuse Process

dealing with people who are making use of ubiquitous computing technology. In particular it describes the capture, representation and potential re-representation of data from a digital environment and the resources both within and outside that environment.

In the next chapter we will look at two systems, *Replayer* and *Digital Replay System* which purport to conform to these guidelines and provide the tools necessary to allow ethnomethodologists and other qualitative social scientists to successfully observe, and unpack interaction, communication, and collaboration with, around and through ubiquitous computing systems. The use of these systems will then be explored in two case studies, before they are compared and contrasted and conclusions are drawn.

## Chapter 3

# Replay Systems

### 3.1 Introduction

In the previous chapter we outlined some core requirements for systems to support qualitative analysis of ubiquitous computing systems. Namely:

- Tools that allow viewing of system log data synchronized with other types of media
- Tools that enable researchers to extract multiple sources of information from recorded logs, and which allow them to edit extracted information and combine it with media from external sources to produce unique datasets.
- A replay system that exploits time stamps to coordinate the use of the multiple media in a dataset, which enables cross referencing and indexing to support the splicing together of multiple media, and which enables multiple media to be played side-by-side.
- Tools that support the production of representations from datasets and which preserve the relationship between representations and the media from which they are derived, and which enable source media to be recovered and viewed.

In this chapter we will explore two systems which approach the task of replaying heterogeneous data from two distinct directions, but both of which aim to fulfil those requirements. The chapter aims to give a broad overview of the functionality of each system, with more specific technical discussion of key features in the proceeding chapter. The first of these systems, *Replayer* approaches the challenges focusing firmly on the system log data and various ways to represent that data and combine it in a synchronized manner with other media such

as video and audio. In particular Replayer provides a set of tools by which once a log has been imported to its internal framework, virtually any log data can be explored through a variety of visualizations, and different views can be used together in synchrony, allowing synchronization to be applied across many different dimensions. The second tool described: *Digital Replay System (DRS)*, Approaches the challenge in a different manner. Because DRS was developed with a participatory design approach [12], working directly with several qualitative social scientists from different fields, it provides support for more methodological synchronization, such as transcription, coding, annotation etc. Log files in DRS have an equivalent precedence to any other media type reflecting their relative importance in the eyes of the intended user group.

## 3.2 Replayer

Developed at the University of Glasgow, Replayer first appeared as a project specific tool for the study of the mobile game *Treasure* [14] combining a detailed animated visualization of the state of the system with several synchronized video streams of the action taken from different viewpoints. The success of this representation as a resource for the qualitative analysis of *Treasure* led to the rewriting and expansion of the system into a more generalized toolkit that would allow system logs to be visualized simultaneously with synchronized video and audio playback.

### 3.2.1 Requirements and Design Approach

The initial development of Replayer was purely to study the single case of the game *treasure*, in which the mobility of the subject made observation difficult. Developed simultaneously with the game it presented an overview of the state of the system from a set of system logs. Figure 3.1 shows this early system. The in-house development team also included an ethnographer, who would be tasked with evaluating *treasure*, so it was tailored more or less specifically to his needs. In practice this meant an iterative design process with several versions of the software being produced in quick succession.

As the ethnographer in question used video analysis as a primary resource for his work, and videos were captured of the trials, it quickly became apparent that there was a need to synchronise the video and log data streams. This then was the first real iteration of the Replayer idea. Further discussions lead to a number of requests for specific features

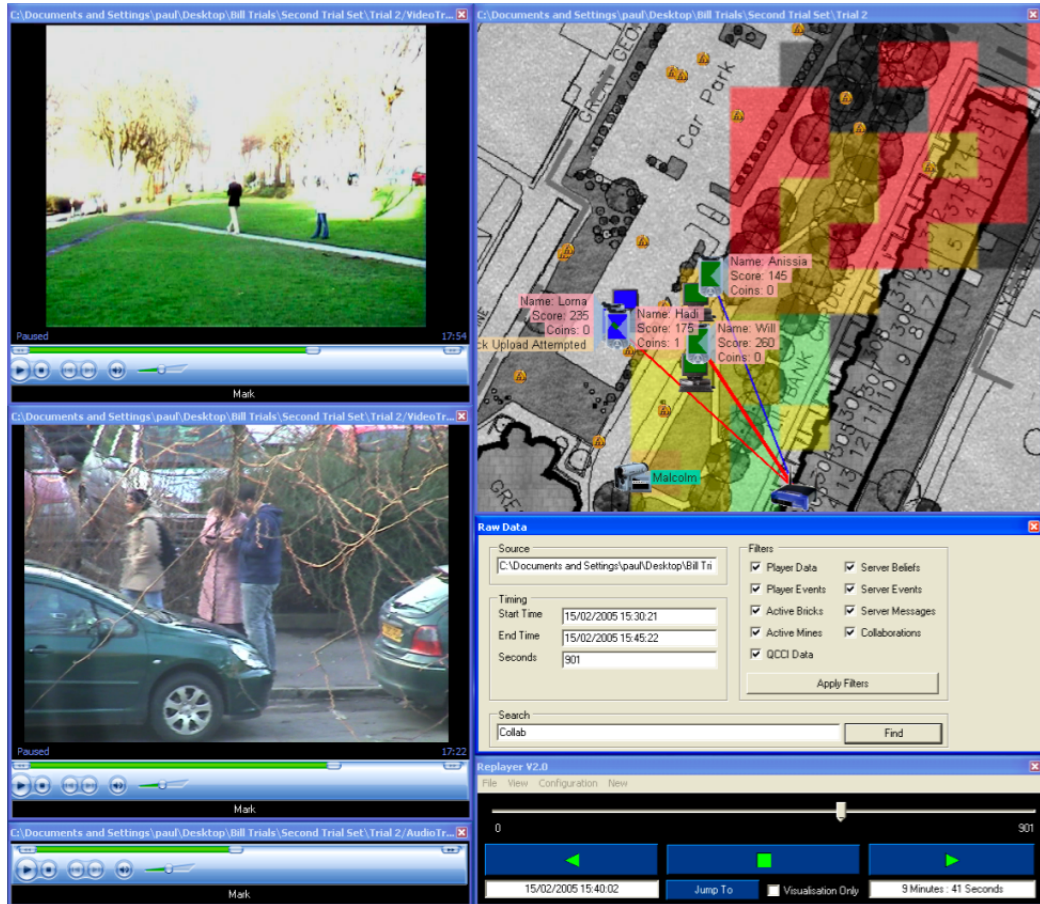


Figure 3.1: An early version of Replayer, specifically designed to analyse data from *Treasure*. On the left we have two videos and an audio stream, right-top we have a live map showing positions of players, connectivity, and some state information, right-middle we have a set of filters for the above map, and right-bottom we have some VCR-like playback controls.

such as counts of events given a particular circumstance, and it soon became clear that a more general-purpose approach was needed. Scaling this up it further became clear that there was a need to generalise the whole system to allow for its use in the study of other in-house ubiquitous computing systems. This led to a general redesign following the one basic principle that visualised log data should be synchronisable with other media such as video. Before beginning to build the system a number of interviews were conducted with HCI evaluation specialists from different backgrounds including ethnography, psychology and more traditional HCI. From these interviews a number of key requirements were constructed:

- Visualise system logs in different ways: graphically, tabular etc.
- Perform some simple statistics on the log data: counts of events given certain circumstances etc.
- Synchronise those visualisations with each other and with other media resources.
- Allow for collaborative exploration of the data: an equivalent of video data *sessions* conducted by one of the interviewees.

These rather general requirements served as a basis to create a first generalised version of Replayer, which was then refined iteratively through its use in the evaluation of several systems including *treasure* [14], *Feeding Yoshi* [21] and *Shakra* [9]. With each system evaluated helping to highlight both usability issues and additional requirements for the software. It is worth noting that the number of social scientists involved was quite small (four interviews and only two of those went on to actually use the system for studies). We will see later the more extensive participatory design approach taken in the development of Digital Replay System. That refined version is the system described here.

Replayer is designed to examine two main types of data:

- Multimedia Data
- System Logs

The first of these, multimedia data, would consist mainly of video and audio files. Replayer would need to be able to play these synchronously with the system logs in the same way that its initial incarnation had. It was a design decision to make direct use of a video playback program for this area of the system rather than building a custom one, as it was

an intention of Replayer to feel like *middleware*, allowing the user to make use of software packages they were already familiar with, for reasons outlined by Paul Dourish in [67]. This was one area where, despite being a cross platform system, a difference between the Windows and OSX versions of the system would have to appear. For windows users, it seemed most suitable to use Windows Media Player, while for OSX users QuickTime would be the familiar playback tool.

The second type of data Replayer would need to focus on was system logs. When discussing system logs in this context we refer to anything recorded by a computing system during its execution. The number of possible data types here are too numerous to completely categorise; indeed that would be an axiomatically impossible task as any new system might record something that has never been recorded before, but examples from past experience included such things as location data, network traffic, signal strengths, system state data, interface events, memory usage and many others. An internal database structure was thus required which could handle any type of recorded data and make it easily accessible and visualisable. A set of viewers designed for displaying data in different ways would be created for visualising that data.

We will see in detail in the next chapter why it was a design goal to create Replayer using a distributed system architecture, but it is enough for now to know that Replayer was designed to be an analysis toolkit capable of handling system logs and multiple multimedia streams that was both cross platform and networkable. Furthermore it needed to include the concept of brushing and linking [19] in order to maintain selection integrity, and therefore data integrity between different viewers - this means that any selection made in one area of the system should affect the selection (where relevant) in all the other areas.

### 3.2.2 The Server

At the heart of the system is a server which must be running for any of the viewers to work. The server has very little in the way of user interaction, beyond some simple output of the state of the database, and self-logging. It is responsible for management of all connected viewers and handles all database transactions.

### 3.2.3 Clients

Users do not as a matter of course interact directly with the database server. Instead that interaction takes place through a management interface called, for legacy reasons,



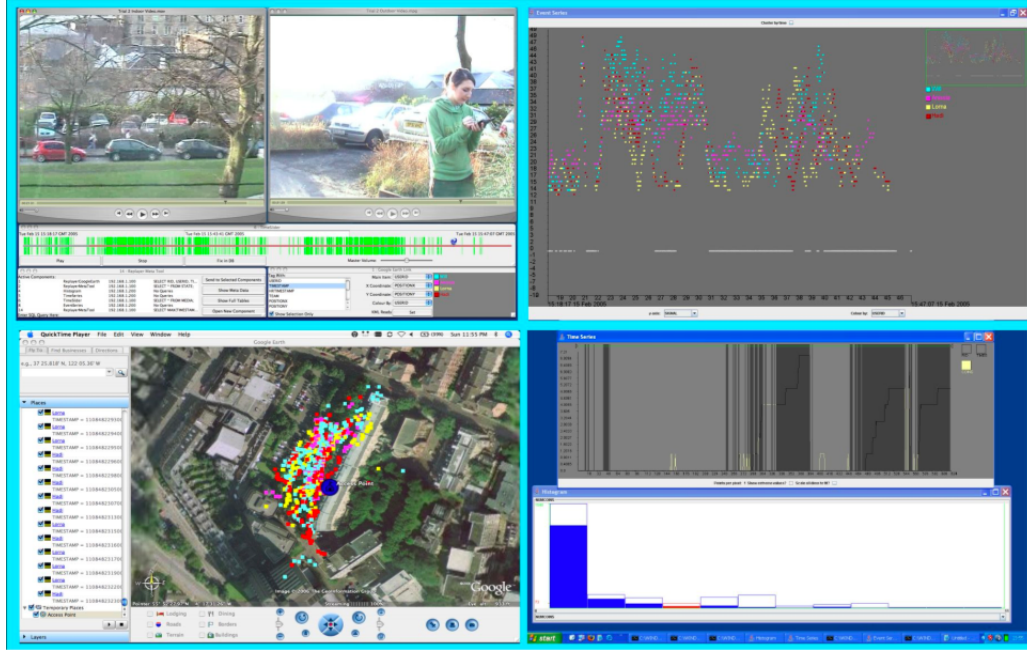


Figure 3.2: Five Replayer tools in coordination. Clockwise from top left, the figure shows the video component handling two synchronised simultaneous videos, the event series charting signal strength for each user over time, a histogram and time series showing summary information on a system property and the map showing the recorded positions of users based on GPS. Data are taken from the *treasure* game.

the *Meta Tool* (figure 3.3). The meta tool serves a number of purposes and unlike the server is replicable within a session - this means that different computers sharing the same Replayer session are able to have a meta tool each. It is for this reason that the meta tools are separated from the server. First, the meta tools display a list of the current active components on the session, along with the name of the computer on which they are running. Behaving a bit like an operating system task manager it is possible to open and close components from here.

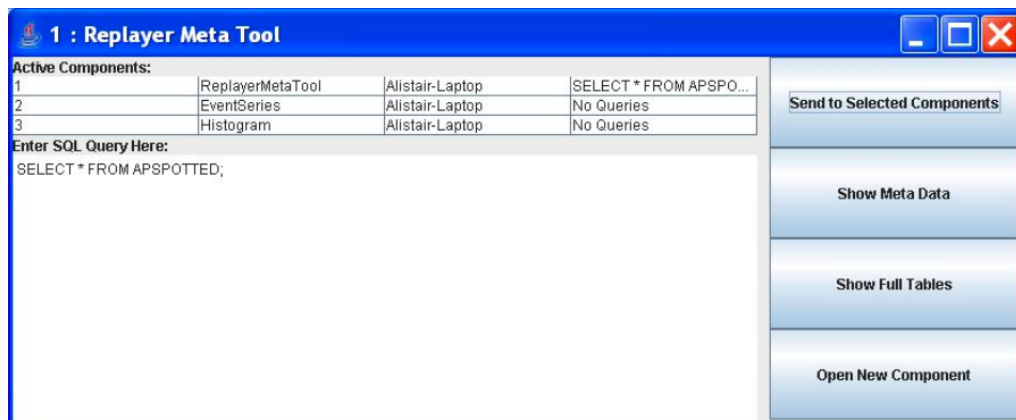


Figure 3.3: Replayer's *Meta Tool*, Used for component management.

The list of usable components is dynamic allowing new components to be added at run-time. To qualify to be a component the program must be packaged up as a java archive resource (JAR) and added to the folder containing Replayer’s executable files. It is possible for these components to be written in any language as long as the executables are subsequently packaged up into jars.

The meta tool is also more importantly, the user’s gateway to the database. It is here that we see an example of Replayer’s versatility, but also its necessary complexity. Components themselves do not directly query the database, as they have no preconception (unless hard coded in application specific components) of exactly what data they require to display. Instead the user must query the database and send that data to an open component. And thus comes the dilemma. The most effective and powerful way to query a database is using the Standard Query Language (SQL) this is a long developed and powerful approach to data extraction, however it is a scripting language which requires some understanding of database theory, or at the very least some practice. We knew it would be difficult to convince non-technical users to write SQL, but it provided an effective means for us to try out ideas until we developed a visual programming approach to writing that SQL. Some understanding of the way it works remains necessary for making advanced queries of the database in the current version of Replayer. The visual programming approach offers the user slightly more to work with. A menu is displayed showing a list of available tables. When a table is selected a conditional panel is displayed. Depending on the type of the column this will take one of two forms. If the data are numerical, a double ended slider is displayed with a histogram directly above it (figure 3.4)., this is based on a similar visualisation component from HiVE [180].

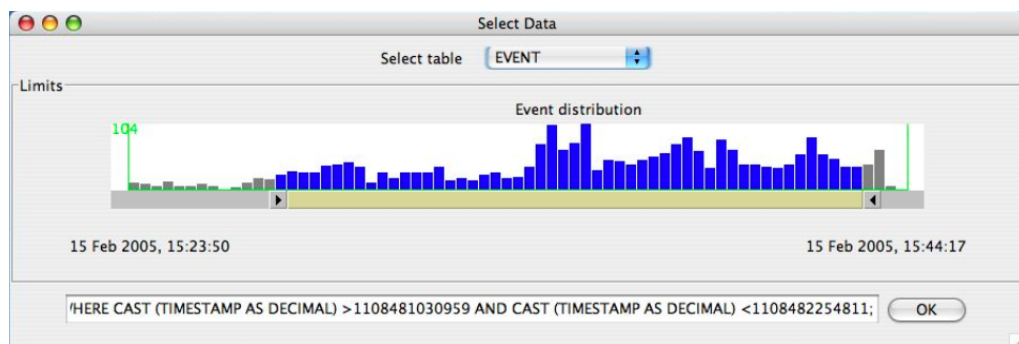


Figure 3.4: A visual interface for selecting data from the database, featuring a double ended slider, with an attached histogram showing the distribution of the data.

This gives the user an overview of the data and allows upper and lower limits to be set.

Alternatively, if the data contains strings, a list of check boxes is displayed, allowing the user to select those nominals she wishes to include in the returned data.

The visual programming tools offer nowhere near the versatility of pure SQL, while SQL alone is simply too complex to be acceptable to non-computer scientists as more than an expert option within the toolset. This is a dilemma that persists throughout the development of both Replayer and DRS. Increasing functionality necessarily increases complexity, which decreases the understanding and subsequently the uptake rate amongst new users. It is important to maintain an awareness that the intended user group are qualitative social scientists and *not* computer scientists though their willingness to experiment with CAQDAS software has been demonstrated in the literature review chapter. A discussion with Greater Manchester Police over potential uses of DRS within their forces yielded an interesting revelation called *the five click rule* - if a user cannot get to where they want to be within five clicks, the system is too complicated. This is probably a good rule of thumb for user interface designers to keep in mind, predicated as it is on the *seven plus or minus two* rule of working memory [195], especially when developing for communities outside computer science. As such, a system requiring the user to understand and write SQL (misuse of which can lead to the compromising of data integrity) is almost certainly over complicated.

### 3.2.4 Log Data Viewers

Replayer provides a number of visualization components for analysing logged data. Each of these receives a table of data from the server, following a request made via the meta tool and displays it in some coherent manner. The intention was to create a set of generically applicable viewers, mostly taking the form of charts of various types. This was never intended to be a complete set covering every possible technique for graphical data analysis, but rather a collection of basic tools which could be extended as required. The coordination of multiple views is supported to allow greater insight to be made into the captured data. This is achieved through a technique called brushing and linking [19], 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, and being able to select this region and immediately see how the same group has been handled by complementary viewers can greatly increase an analyst's understanding of the inherent structure of the data.

### Time Series Viewer

The first tool incorporated into Replayer is a *time series*, designed to plot various types of temporal data. 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.

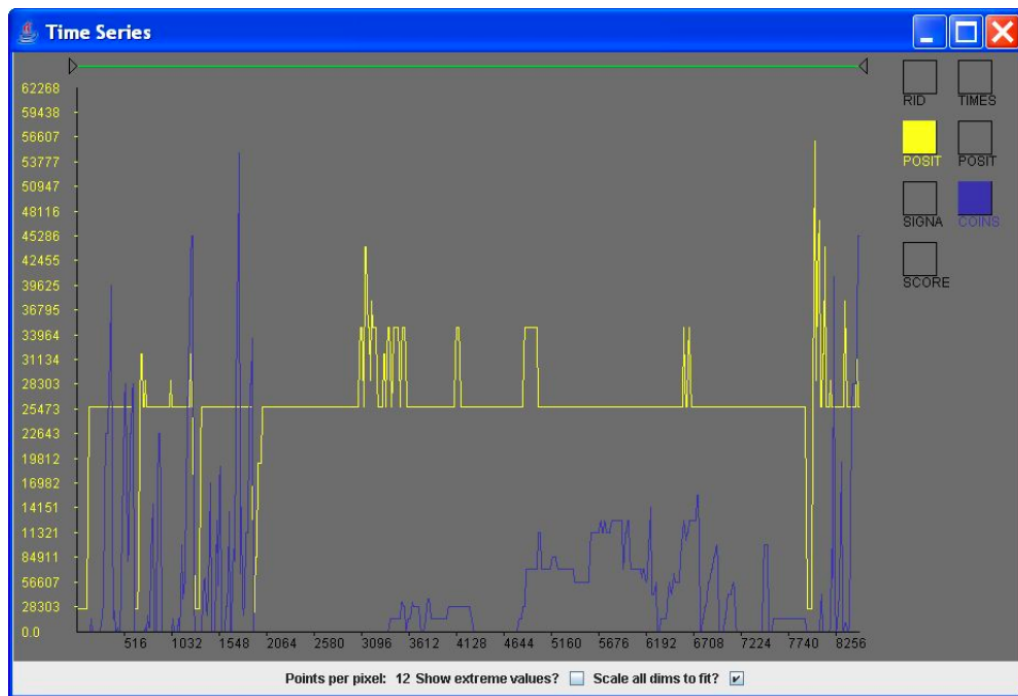


Figure 3.5: The *Time Series* component plots logged data over time. The tool can compare attributes between users or view a summary of a single users' performance

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 3.6, the time series is reflecting selections made in another component. The time series variables are greyed out at the deselected periods, with the background also darkened.

### 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 are 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

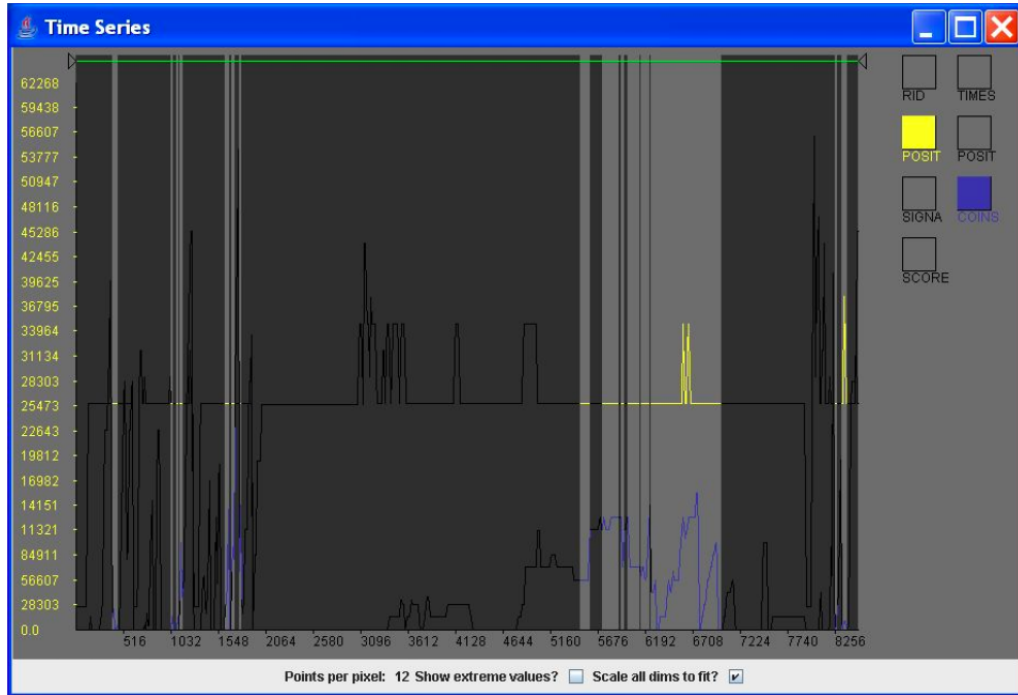


Figure 3.6: The time series has received a remote selection and has shaded the de-selected periods.

discrete events that happen occasionally 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 3.7 shows an example, displaying data recorded over a week-long system trial of the mobile game: *Feeding Yoshi* [21]. 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 set of events, 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 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

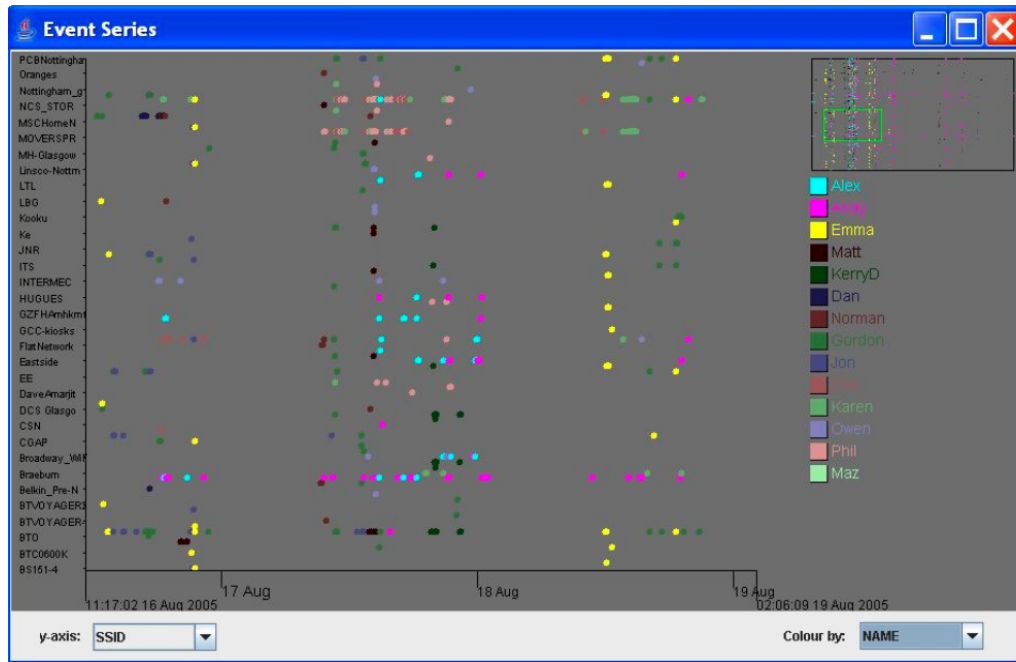


Figure 3.7: The event series tool plots discrete events by time. In this example, the y-axis has been set to display owner-assigned names of wireless network access points, and each event is coloured by user name.

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. Figure 3.7 shows a selection of SSIDs: (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.

The Event series viewer also has some additional advanced tools, including support for single link clustering of events, and multiple selection comparison (via the histogram).

## 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 3.8 illustrates.

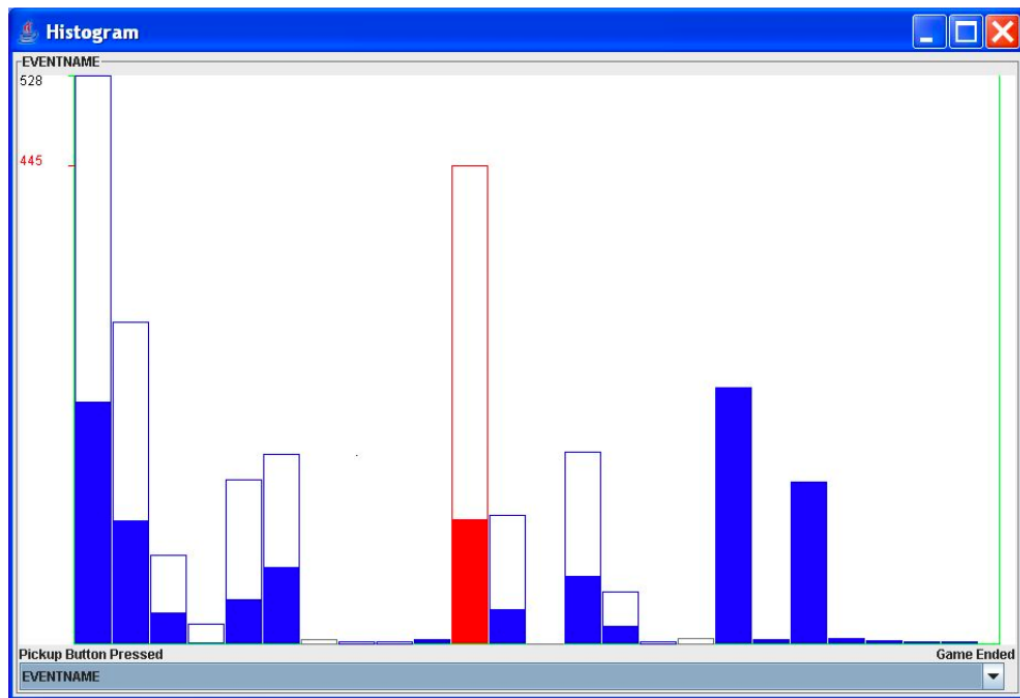


Figure 3.8: The histogram component shows a distribution over time. This example shows the occurrence of each of a number of events. Bars are filled in proportion with the number of each such event that exists in the selection.

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. 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.

### Graph Drawing

Replayer includes a graph drawing component. In this case we mean a graph in the mathematical sense, as an abstract model used to represent data consisting of inter-connected objects. Through an iterative force-modeling process [87], a drawing can be constructed to convey weighted pair-wise relationships between a large set of objects.

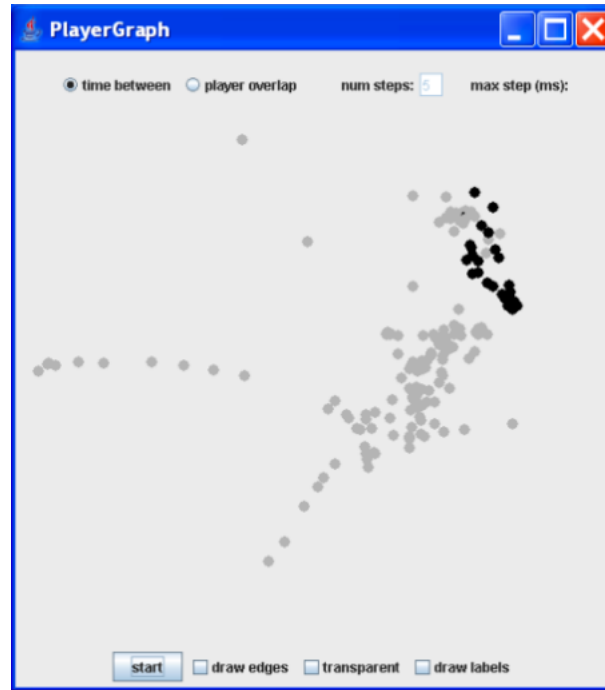


Figure 3.9: Replayer’s Graph Drawing Module showing a selection of events that happened in a short space of time, with clustering based on the amount of time between events.

This is one of the more abstract of Replayer’s components, designed to cluster sets of data in meaningful ways. One example of its potential use is to recover spatial information where geographical log data are not available - as in figure 3.9 where a quasi-spatial positioning for a given user over a short period of time is being reconstructed from logs containing no location data. Clustering between the vertices (i.e. the length of the edges) is based on the average amount of time between repeated events - in this case repeated accesses of certain wireless hotspots by users. So the result is a topographical map, that while it may bear no resemblance to the physical layout of the hotspots, may still provide some useful information. We need only look at the famous London underground map by Harry Beck (1933) to see a fine precedent for the use of topographical mapping.



### 3.2.5 Statistical Analysis Tools

Replayer provides a small set of tools for doing certain kinds of statistical analysis. It was decided early in the development process that complex statistics were outwith the design goals of the system, it would have been pointless to replicate all the facilities provided by packages such as the Statistics Package For Social Science (SPSS), especially as this is a package that most social scientists are already familiar with. Therefore Replayer simply allows exporting to a format readable by that package. However, some basic numerical analysis tools are provided including a simple SQL based tool for counts, summations and averages, a correlation graph tool, and a tool specifically created for examining mutual information relationships.

#### Basic Stats Tool

Another in a long line of Replayer misnomers, the *stats tool* is actually a visual implementation of SQL's math functions. It generates the SQL statements required to achieve simple functions such as SUM, COUNT, AVERAGE and many of the other options provided within the SQL language. The output is displayed on a table, and can be routed to the other components where appropriate. As with the meta tool, the stats tool leaves the SQL visible, allowing the user to edit it offering greater versatility for those familiar with that language, but is structured enough to be at least possible to use without prior SQL knowledge.

#### Correlation Graphing

One common practice in qualitative analysis is performing correlations between two types of data. The tool takes two streams of data, generated by two SQL statements, or routed from other components and displays the results on a simple x,y graph.

Like the other Replayer components it supports brushing and linking, but unlike the others, the data displayed is not actually directly from the database, instead being the product of two values. However, making a selection in the correlation tool results in the values used to create that value being selected in the other components. This presents an interesting idea. Much of the practice of statistics involves creating levels of abstraction. Once that abstraction has been applied, it is difficult to get back to the earlier levels. With Replayer's brushing techniques, it is simple to see the reasons behind outlying data - imagine a case where we have an apparently good linear correlation between two data streams, however we have one value lying seriously outside the line, and drastically throwing off the

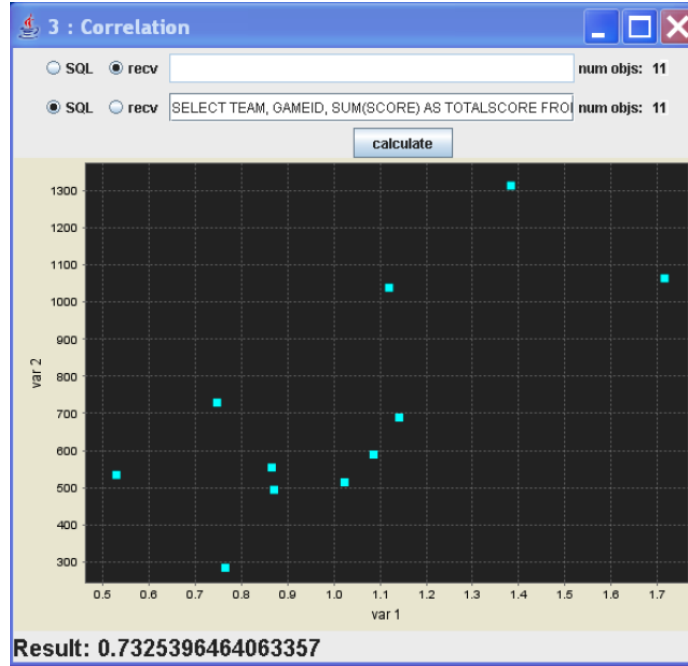


Figure 3.10: Replayer's Correlation Component

result. In Replayer we can select that value then examine the original data leading to that value and discover the reason for that outlier - it may be valid, or perhaps it is the result of some system error. Whatever the reason might be we are able to reverse our abstraction to examine the underlying data with a single click, along with related contextual data, even if those contextual data are found in something more traditionally qualitative such as a video.

### Mutual Information Tool

It is possible to use Replayer to do more complex numerical analysis. A special component was created primarily as an example of the kind of sophisticated analysis that could be possible within Replayer were more of such components created. This component is designed to examine mutual information between two data streams. Information theory is a mathematical study of the encoding and communication of information, which provides several measures for the calculation of dependencies and relationships between data sources [194]. One such measure that is of use in this work is mutual information (MI); a property used in considering the independence or interdependence of two variables. It measures the amount of information that can be gained about a variable  $X$  by knowing about another,  $Y$ . A Replayer component has been constructed to perform mutual information calculations between streams of recorded data. This can be applied to any timestamped numerical measurements,

allowing an analyst to assess the relationships between the various recorded streams.

### 3.2.6 Google Earth Bridge

As has already been noted, one common feature of ubiquitous computing systems is location data. It is thus important to provide some form of graphical display of recorded geographical data. Google Earth<sup>1</sup> is able to display custom data in the form of Keyhole Markup Language (KML) files, which can be retrieved from a network server on the basis of regular updates, which combined with its comprehensive mapping facilities and widespread use made it an excellent resource for Replayer to exploit. Figure 3.11 shows Replayer's Google Earth Bridge component.

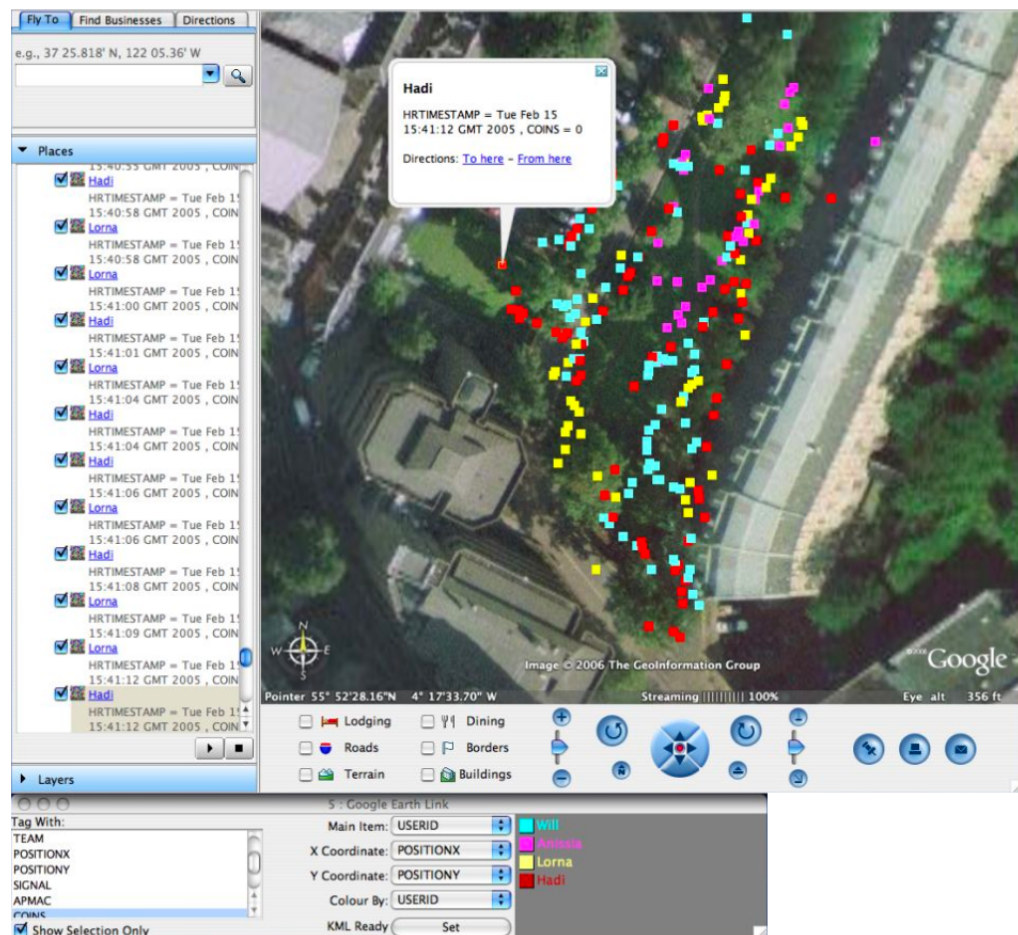


Figure 3.11: Replayer using Google earth to create a spatial distribution of events for each user. Top: the normal Google earth application. Bottom: Replayer's bridge to Google earth allowing control over which events are displayed and how.

This component serves as a translator between Replayer and Google earth, generating

<sup>1</sup>earth.google.com

KML files and serving them to Google earth via an http connection. The HTTP GET command from Google earth includes the coordinates of the four corners of the rectangle of map currently being displayed. This means that selection can be achieved by zooming the rectangle to cover a subset of the data.

### 3.2.7 Multimedia Components

The Replayer multimedia components make use of different playback software for each platform (Windows Media Player for Windows and QuickTime for OSX). Both of these players are controlled in a similar fashion to the Google earth component described above, using special bridging components. The interfaces to these components are identical, with only the internal implementation and ultimate result changing. These are duplicate components so as to have specific versions for specific platforms. The user interface to the bridges are written in Java, while the QuickTime component uses AppleScript to control QuickTime player on OSX based systems, and the Windows Media Player component uses C# and the common object model(COM)interface to achieve the same results on Windows-based computers. The initial visualisation presented to the user is that of a Timeline component (figure 3.12). In the example shown in the figure 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.

The start time and duration of media clips are stored in the server's database. On start-up, the component colours on its timeline all the places where media is available. This alone can be important - a system log may potentially last many days or even weeks, and video clips may be sparse, so being able to locate them on a timeline is a useful initial visualisation.

Using this component, new clips can easily be added to the database, with synchronisation being achieved with the QCCI technique described in the next chapter. The timeline shows the current selection by highlighting the relevant slices of time in green. When a selection is made from here, or from another component, the media clips will automatically jump to the first selected frame. As the thumb is moved in the timeline, the clips move, with

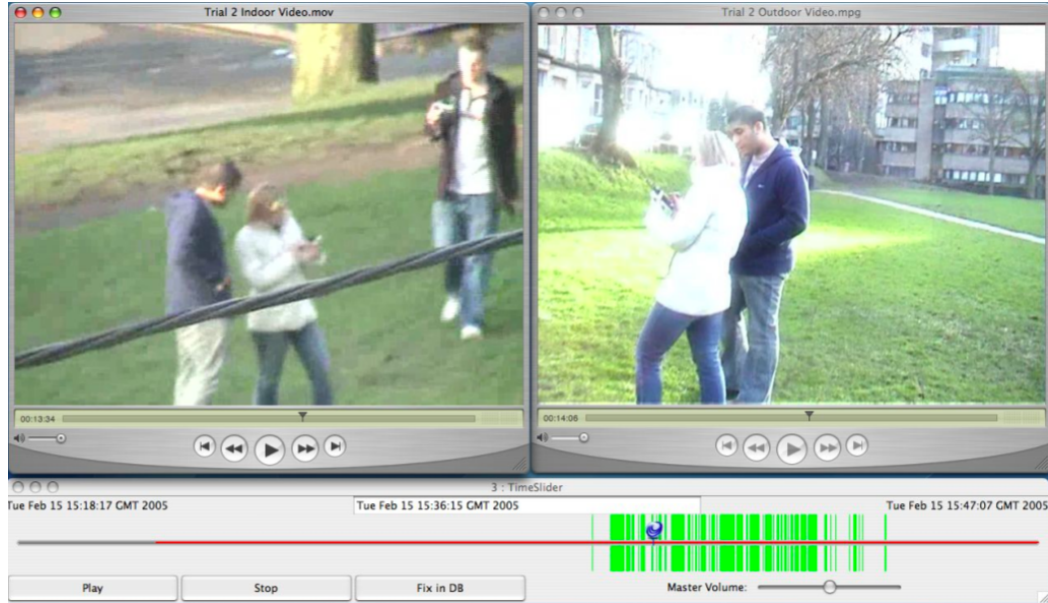


Figure 3.12: Replayer's Time slider component and quicktime videos

only those that are related to that time being displayed. A 'scrubbing' mode is provided which allows the user to drag the thumb, thus changing the frame of the movies and affecting the selection in all peer components in real time. A 'play-all' button is included, which plays all the video clips synchronously and sends relevant selections to peer visualisation components. If no video is available - the play function can be set to jump between times where some data are available, skipping any gaps in the logs. Additionally when using this mode the time delay between frames can be changed, allowing components to be animated faster than real-time.

### 3.2.8 Summary

We have seen some of the functionality of Replayer, as well as some of the motivation for why it was designed in this manner. Replayer provides tools to store and access data on a local database server, and run a managed set of visualisation components, including time series, event series, histograms, maps, video etc. Each of these components allows a different way to explore the data, and they can be used in coordination either simply by synchronised playback, or by the shared selection system (brushing and linking).

### 3.3 Digital Replay System (DRS)

The *Digital Relay System (DRS)* is an application developed by The Digital Records for e-Social Science (DReSS) Node at Nottingham University. DRS is a cross platform, Java based analysis toolkit that supports the collection, collation, storage, markup and representation of digital records. Unlike Replayer, DRS has all its tools integrated into a single program. The layout is designed in a similar manner to the Integrated Development Environments (IDEs) used for programming, with some areas resembling those of video editing suites.

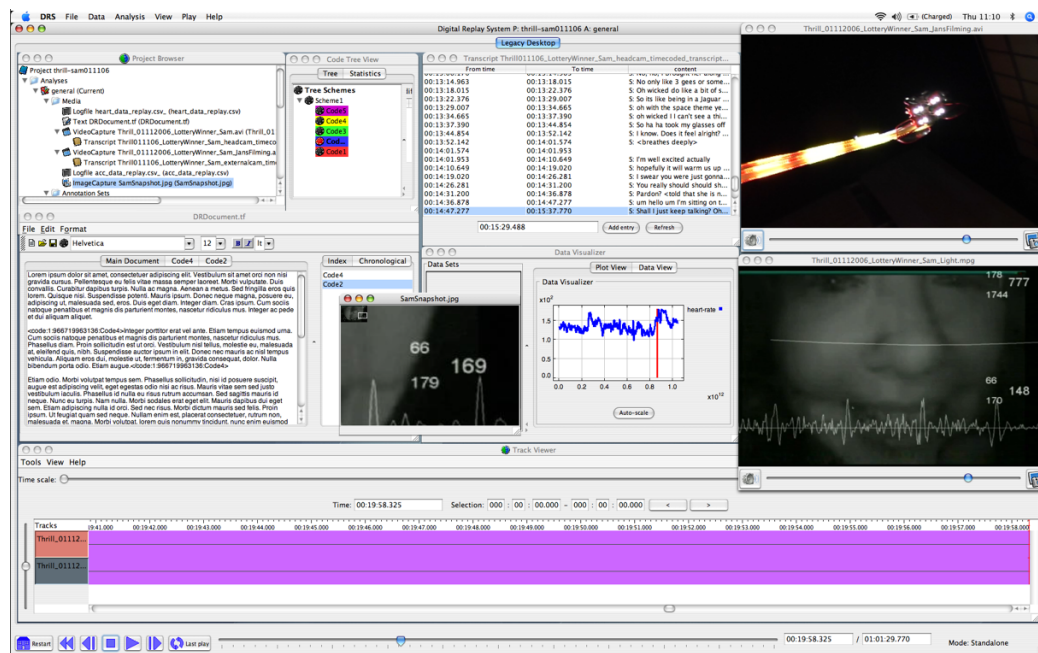


Figure 3.13: Screenshot from DRS

Like Replayer, DRS is prone to use a lot of screen real estate, but unlike Replayer this cannot be solved by opening up additional viewers on networked computers, so instead familiar window organization menus are provided. Generally DRS must be used in multi monitor setups, or at least on high resolution monitors, to allow for the necessary space to lay out several coordinated viewers. However, as a consequence of having a containing application, the whole of DRS can easily be minimised - something that was not feasible with Replayer. While that sounds like a minor issue, in terms of usability it is actually quite a serious one.

### 3.3.1 Requirements and Design Approach

The design and development process of DRS differs significantly from that of Replayer. Developed as part of the work of The *Digital Records for e-Social Science* research node at the University of Nottingham, the approach taken was somewhat different. The research node is an interdisciplinary collaboration between computer scientists, ethnographers, psychologists and corpus linguists providing a wide range of social science professionals with whom to conduct a strong participatory design approach. The design has been forwarded by a series of *driver projects*, one from each discipline, each with its own specific requirements. The first, based in the Department of Computing Science is concerned with supporting the process of ethnography and thus relates to the development of tools to directly support observation and description, including the exposition and visualisation of system logs. At the time of the driver projects this was largely concerned with studying the use of several mobile games developed by the Mixed Reality Laboratory at the university of Nottingham. The second project, based in the Department of Psychology focuses on studying the use of a system called VIRILE [34] designed to teach chemical engineering students about the practicalities of working in a plant. The study is concerned largely with the markup of recorded data and has requirements relating to the synchronisation, coding and application of meta data to recorded data, but also makes use of the system logs to describe the *state* of the system at any given time. The final project, Based in the Department of English Studies, is concerned with the construction of multimodal corpora, extending the more traditional textual corpora to also handle video allowing for the study of the rich context and back-channelling associated with the use of language in situ. It makes use of logs generated by a gesture recognition system to automatically generate codes defining particular gestures and collocates them with particular language utterances. A number of corpus and computational linguistics approaches are then applied to the created corpus [133]. In a series of focussed design meetings a number of key use cases were explored leading a set of core requirements for the system. These core requirements were as follows:

- Must be cross-platform
- View and synchronise media of various types (video, audio, images, documents, transcriptions etc.)
- Organise a corpus of heterogeneous media.

- Mark up media (transcription, coding, annotation).
- Import and visualise system logs in various formats, synchronised with other media.
- Perform concordance and other text-search techniques over annotations transcriptions and codes.
- Share corpora with other users

An initial prototype was then developed. This was presented to each of the driver projects who then applied it to their own project data and returned with reports about usability, feature requests and additional use cases. This process was repeated several times until a general public release was made available. Next a collaboration was formed with the CAQDAS group at the University of Surrey whereby they promoted the tool to a wider community of social scientists. Working closely with CAQDAS as well as the original driver projects enabled the further refinement of the system, and a second public release was made. As of December 2009 the webstart version of DRS has been launched by 1542 unique IP addresses, and the source code has been downloaded by 638 unique IP addresses.

The direct collaboration in development with exactly the target user group has allowed DRS to develop into a practically usable toolkit supporting a wide variety of qualitative social science practices. This method of participatory design, along with iterative prototype development has been exploited to develop a tool more directly suited to the needs of the community than Replayer, while still providing the innovative key features for observation. However, because of the nature of its driver projects, it is mostly focussed on the study of ubiquitous computing systems, and its features have been designed with that task in mind.

### **3.3.2 Corpus Management**

DRS provides the concept of projects and analyses as a method for the collation of one's data. Unlike Replayer which could store only one dataset at a given time, DRS allows its users to store all of their data permanently within the system. The data are then split up into a series of projects, with each project split into a series of analyses. This is somewhat more in line with existing tools outlined in the related work section, and marks DRS as more of a plausible tool than proof of concept - though it is still very much a prototype.

A project in DRS refers to the repository into which all the media and derived media for a particular study are stored. It is possible to have as many projects in the main DRS



database as one wishes, with each one related to one specific collection of data. This allows completely distinct experiments to be stored concurrently. Within a project is stored all the associated media as well as derived media such as coding schemes transcripts etc. Also stored in a project is associated metadata such as information about people and devices related to the project. Data are never actually deleted from a project, but simply marked as deleted, and displayed in a recycle bin. In actuality, the media is not itself stored within the project but rather a dynamic link is stored within an internal resource description framework (RDF) [201] based representation.

An Analysis represents a single study over a subset of media from a project. It contains all the media and derived media associated with it. It also contains all the time synchronisation data for those files - this means that media files can be synchronised differently for different analyses. An analysis depends on a parent project. Once media has been added to an analysis it is available to be synchronised and viewed with one or more of DRS's viewers. Depending on the type of media, a number of context sensitive options become available, including options such as import/export, transcoding and application to other areas of DRS such as the *Track Viewer*. Unlike a project, which serves simply as a repository, and analysis has a concept of time and synchronisation. It is therefore possible once in an analysis to simultaneously view several synchronised media. Media playback is controlled within the DRS environment only at analysis level.

### 3.3.3 Time and Synchronisation

Each analysis has a master timeline across which all other time-based media should be synchronised. Each item of time-based media or derived media can be considered to have a duration which must be placed somewhere on the master timeline. Figure 3.14 shows the way in which several media files can be positioned on the master timeline. Note that Video II and its associated transcript are locked together, so that moving one will affect the other. Synchronization information is defined either by directly inputting the numerical offset information to the dedicated synchronization editor tool or by visual manipulation of tracks in the *Track Viewer* (see below).

### 3.3.4 Media Files

DRS deals with two particular types of media: *time-based media*, and *discrete media*. Time-based refers to anything which has a duration such as audio, video etc. Discrete media refers

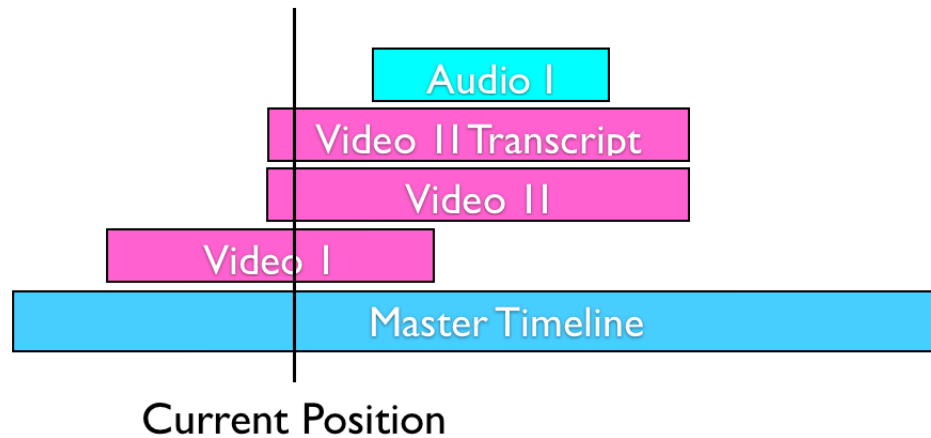


Figure 3.14: Positioning media on the master timeline.

to those media-types in which time is not a meaningful thing, such as images. Additionally, DRS generates special media types called *derived media*. These are not physical files stored on the disk, but rather descriptions such as annotations, coding tracks and transcripts which depend on another file in order to make sense. These derived media files exist only within the DRS environment, though each can be exported into a physical file if required.

When importing media files into DRS, some information has to be provided to inform DRS how to store them - this allows DRS to apply its ontology to the media. This information typically includes the filename and path, a title, by which the file will be referred to within DRS, the mime type and finally the media type.

### 3.3.5 Media Viewers

The different types of media supported by DRS necessitate a variety of ways to view them. Opening a media file from within the application offers a filtered choice of viewers which can be used to examine that particular media type.

#### The Video Viewer

The *video viewer* (figure 3.15), as the name implies is used to display video files. It is possible to have any number of video viewers open simultaneously, however playback quality will reduce as the number of viewers increases. The number of viewers that can be displayed is highly dependent on how powerful the computer is. The viewers are based on QuickTime

for java,

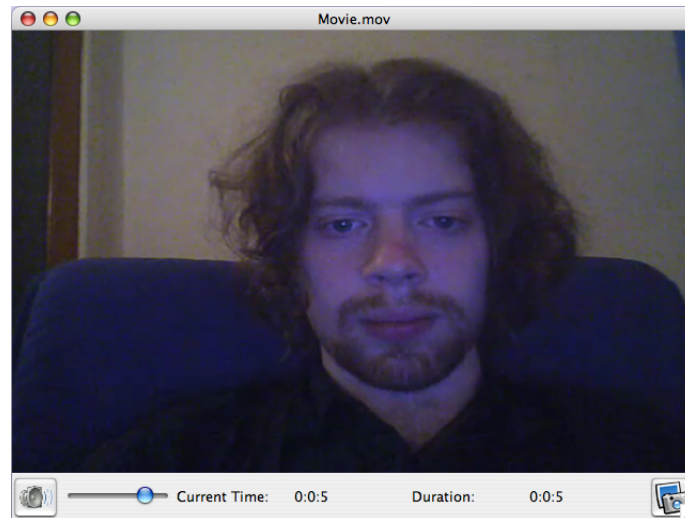


Figure 3.15: DRS's Video Viewer

The viewer as shown in figure 3.15, is based on QuickTime, so can playback any QuickTime supported video codec. This which means that playback is only limited by which codecs are supported by the installed version of QuickTime.

There is no provision in DRS to play a single video without playing the main timeline, playback is exclusively controlled using the VCR controls at the base of the application. Individual movies have independent volume controls however. Also displayed is the current time within a given movie (note that this is different to the current time within the Analysis displayed at the bottom of the screen beside the play controls, because individual moves can be positioned across the main timeline (as shown in figure 3.14). The Duration of the video clip is also displayed.

The video viewer supports the ability to capture specific frames and store them as images by means of a frame-Capture button, which allows individual frames to be quickly exported as jpegs, which are then automatically imported into the analysis. This feature exists primarily to support the representation process, so that as one creates a story, one can use specific frames as evidence, that can be included in a description.

### The Audio Viewer

The peculiarly titled *audio viewer* (figure 3.16) provides a method for playing back audio files within the DRS environment. It takes the form of a window with a volume control and a basic graphic equaliser. The audio viewer supports only two codecs, WAV and MP3. Like

the video viewer above, individual playback controls are not available - playback is *always* project -wide, and controlled with the VCR Controls.

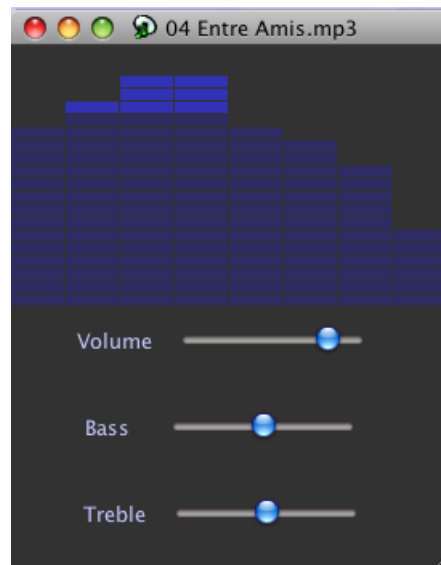


Figure 3.16: DRS's *Audio Viewer* plays MP3 and WAV files.

This viewer serves as a playback device, with a simple volume and graphic equaliser provided to support better audio analysis, but the audio file is perhaps more usefully viewed as a track in the track viewer (figure 3.17), in which the wave form is displayed. This feature is primarily provided to support transcription (in a similar method to Transana - discussed in the related work section 2.4.2) and annotation, as it can help with the correct temporal placement of these annotation objects.

### The Image Viewer

The image viewer (figure 3.18) is a simple tool for examining images in one of the common forms: JPEG, PNG or GIF. It provides support for zooming and panning and is capable of displaying very high resolution images such as those taken by high quality DSLR cameras.

The image viewer also provides a context view, which shows the whole image and a small white rectangle showing how much of the image is displayed in the main viewer.

It is possible for several images to be viewed in a sequence, with each image having an associated timestamp. This means that images can be 'played back' along with other media, with the displayed image changed as the playback progresses. A filmstrip-style view is provided at the base of the viewer allowing users to jump to specific images, and subsequently jump the position in the main timeline. This is in line with DRS's nod to the

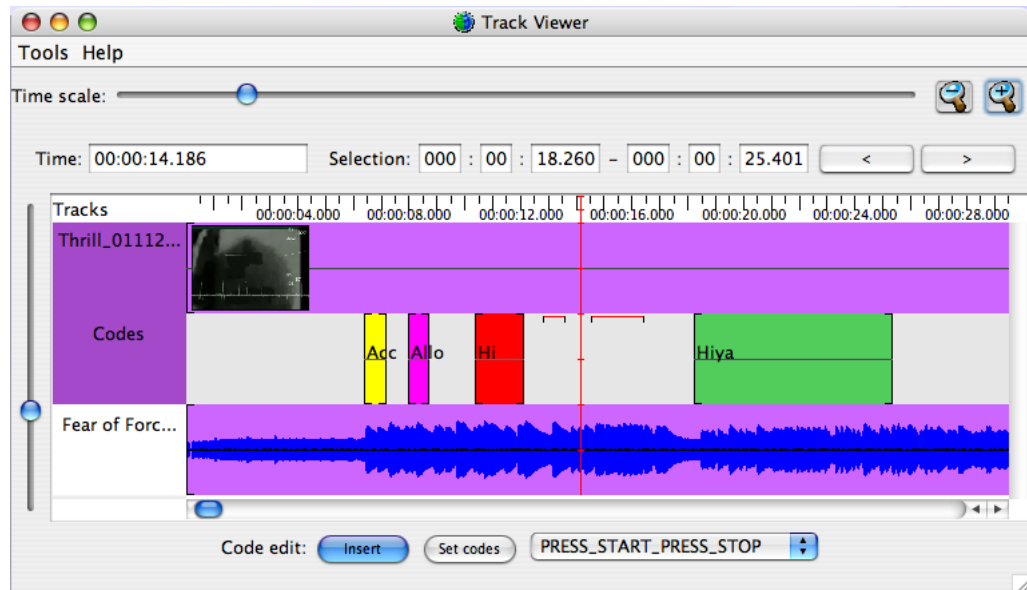


Figure 3.17: The *Track Viewer* showing three ‘tracks’ - one video (top), one audio(bottom) and one coding track(middle).



Figure 3.18: DRS’s Image Viewer

interfaces of video editing software such as Adobe's Premiere <sup>2</sup> and Apple's Final Cut Pro <sup>3</sup>.

### The HTML Viewer

A very simple HTML viewer is provided to allow users to view documents in HTML or plain text. It is non editable, but provides support for examining off-line web pages, which can form a resource for analysis like any other media type.

### 3.3.6 The Track Viewer

The *track viewer* (figure 3.17) is a versatile tool which provides a visualisation of any item or collection of items of time based media in an analysis. Each media item is associated with a track which is displayed on the viewer. A red line shows the current time, and clicking on a track will jump the analysis time to the position clicked. The time is also reflected in a text box at the top left.

#### Track Types

Each of the time based media can be displayed on the track viewer. The supported media types are:

- Video - displayed as a solid purple bar.
- Audio - displayed as a waveform to support synchronized transcription and annotation.
- Transcriptions/Free Annotations - Appear as boxes on the track. Most of the text is hidden unless the current analysis time coincides with the annotation. The process of transcription will be covered in greater detail presently.
- Coding Tracks - Appear as boxes coloured with the code's colour. Coding and coding schemes will be covered in greater detail in a later section.

#### Other functionality

The track viewer can be horizontally zoomed indefinitely allowing the granularity of the tracks to be changed. It is also possible to zoom tracks vertically simply to make better

---

<sup>2</sup>[www.adobe.com/PremierePro](http://www.adobe.com/PremierePro)

<sup>3</sup>[www.apple.com/finalcutpro](http://www.apple.com/finalcutpro)

use of available space. This allows for very fine grained manual synchronisation of tracks as they can be dragged around on the timeline from here.

It is also possible to limit DRS's playback to a specific slice of time. Something that is frequently required for transcription. the user can by using context menus for the tracks, set the start and end times limiting the playback range. The user will see the slice of time selected highlighted and the playback will loop or loop back and forth as selected.

### 3.3.7 Transcriptions

A transcript is a form of derived media. It refers to a list of all the unique utterances, along with the time they occurred, from a time-based-media file such as a video or audio recording. Each transcript is usually associated with a time-based-media file, meaning that they begin at the same time when playing in viewers. Transcriptions are usually either created in DRS using the transcription editor, or imported from Transana.

Transcriptions, and free annotations generally, form an important part of many qualitative analyses. The actual uses of them will differ based on the specific project, but the need is almost universal. While many transcription packages have the option of time synchronisation, Only DRS allows the user to share that synchronisation with other related media, and this facility supports the process of indexing, in much the same way that Replayer's brushing and linking does.

There are three methods by which transcriptions can be examined or created. These are the *transcription editor*, the *annotation table viewer* and the *track viewer*. Transcriptions are not stored by DRS as external files, as with other media but rather a collection of annotations within a greater DRS project file. Because they are stored as part of the RDF, but appear to be distinct files in the project and analysis browser, this can create a certain amount of confusion - especially when dealing with transcriptions imported from Transana. There were, during testing, a number of cases of users editing the original file then failing to realise that the edit would not carry through to the version stored in the RDF model, which suggests that our users' mental model may not in fact match the reality of how their data is being stored.

#### The Transcript Editor

The *transcript editor* (figure 3.19) is one of two tools created to handle transcriptions, the other being the annotation table viewer. The transcription editor provides two primary

functions creating/editing and viewing transcripts of time based media. When viewing, the transcription element reflecting the episode time will be highlighted in green. This highlight will move during playback to reflect the progress of the conversation.

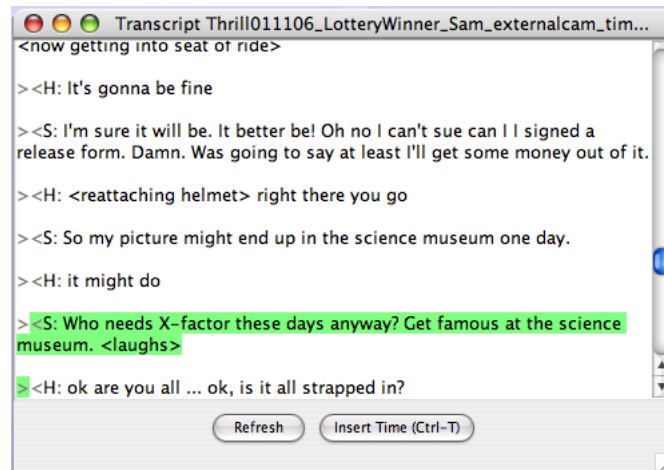


Figure 3.19: DRS's Transcript Editor.

Creating a transcript with the transcript editor is simply a matter of plying back the media that one is transcribing then inserting transcription elements at the appropriate times either by clicking the insert button, or by using the associated keyboard accelerator.

### The Annotation Table Viewer

The *annotation table viewer* (figure 3.20) allows a user to get a tabular overview of any type of annotation track including transcriptions and codes. The annotation table viewer displays the start and end time for each annotation element along with the content of the annotation, all in tabular form.

From time	To time	content
00:10:48.328	00:10:51.444	S: Can't wait to tell every...
00:10:51.444	00:10:57.681	S: Yeah it's absolutely fin...
00:10:57.681	00:11:09.490	S: Hey Sophie
00:11:09.490	00:11:12.604	S: It keeps slipping to th...
00:11:12.604	00:11:22.946	S: Do I have a wonky sha...
00:11:22.946	00:11:26.587	S: OK
00:11:26.587	00:11:36.647	S: Yeah, yeah, yeah can't...
00:11:36.647	00:11:41.870	S: Ok see me now? Hello...
00:11:41.870	00:11:54.746	S: On the building. Oh, h...
00:11:54.746	00:11:57.161	S: Have you been on this...
00:11:57.161	00:12:04.238	S: Did you enjoy it?
00:12:04.238	00:12:08.479	S: Oh wow, how cool!
00:12:08.479	00:12:12.741	S: I'm really cold actually...
00:12:12.741	00:12:16.344	S: <breathes deeply>
00:12:16.344	00:12:20.685	

Figure 3.20: DRS's Annotation Table Viewer.



Hovering the mouse over one of the content cells will result in a tooltip showing the complete text of that element. When the system is playing, the correct element for any given time will be selected, and clicking on any element will jump the analysis (and consequently all the other viewers) to that time. It is also possible to use the annotation table viewer to edit or add new annotations at any given time.

The transcription viewer and annotation viewer provide very similar functionality. The reasoning behind making both of them is that the table viewer makes for a good overview - and was specifically requested by one of the driver projects, while the transcript editor supports more ‘free’ text, which allows users to employ conversational analysis markup [228] - something we hope to directly support in DRS in the future perhaps supporting some form of multi-modal markup as well such as MURML [135], though this latter is probably more relevant to the following section on coding.

### 3.3.8 Coding

The practice of coding is widespread among social scientists, particularly those concerned with formal analytic methods, as discussed in the related work section. It is a technique by which a media file can be annotated with a finite set of codes or annotations which serve to define the subject’s behaviour in some way. Codes are generally described by means of a coding scheme, though the scheme is often subject to change as the task of coding progresses.

#### Coding Schemes

A coding scheme is arguably a form of ontology. It may have many levels of hierarchy. to reflect this structure, in DRS, a coding scheme is defined using the Coding Tree (see figure 3.21 in which a simple coding scheme to define some basic gestures has been created).

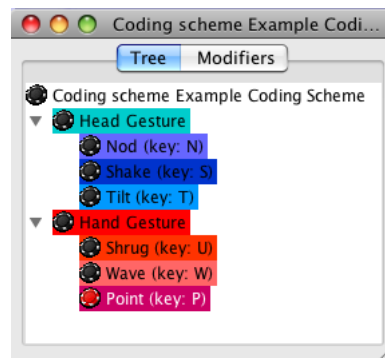


Figure 3.21: An example of a very simple coding scheme.

Any given coding scheme has:

- A name.
- A maximum depth, i.e. how many levels of sub-codes are allowed.
- A timing type, which defines how the codes will be used when coding - see below.
- A nominal duration/period of each code, for certain timing types only (event with nominal duration and state with periodic changes) - see below.

DRS supports five distinct timing types for coding schemes:

- Event with variable duration
- Event with nominal duration
- State with explicit switching
- State with periodic changes
- ‘untimed’, which means that the codes cannot be used to do time-based coding.

In this case events refer to things which happen at particular times, e.g. a gesture or action while states are things which are always present or observable in some form but which may change from one form or value to another at particular times, e.g. whether and how someone is moving (stationary, walking, running, etc.). These cases are illustrated in figure 3.22.

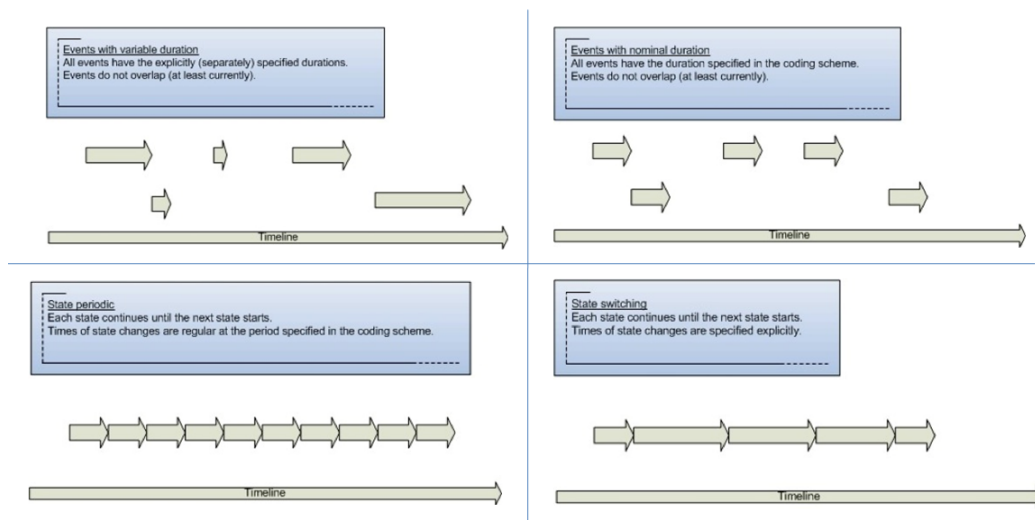


Figure 3.22: Illustration of different coding types.

Further to the general information about a whole coding scheme, each individual code also has certain information associated with it:

- A name (effectively the code keyword).
- A colour.
- Optionally, a key binding, i.e. a key on the keyboard associated with the code for real time coding.
- Optionally, a set of ‘modifiers’, i.e. additional parameters or assignable values associated with the code (e.g. the object of a reference)

### Coding in DRS

Once a suitable coding scheme has been created, the process of actually coding can begin. Coding of time based media is performed using the track viewer (figure 3.17).

Coding is organized into coding tracks (which are a particular kind of annotation set). A coding track can only have one state or event at any given time, so concurrent states or actions must be coded on separate coding tracks. Each coding track has the following properties:

- A coding scheme
- A name
- Optionally, a piece of media which it is specifically associated with (e.g. if the codes are specifically references to things observed in that piece of media)

For most coding scheme types (events with nominal duration and periodic and explicitly switching state) each code needs only one time, as so is simple to insert with one key press (or mouse action). For events with variable duration, however, the end of each event must be explicitly specified, typically by tapping the same key again, or releasing a held key.

Additional (non real time) coding can be performed using the mouse by first selecting a region of the coding track by clicking and dragging on coding tracks.

A coding track - like any annotation set - can also be viewed using the annotation table viewer. This gives a tabular view of event/state start and end times and code (text).

### Automated Coding

The process of coding is time consuming and intensive. In most cases this process is simply necessary, and is a skilled part of performing an analysis, but there *are* some instances when it might be feasible to automate this process to some extent. System logs in and of themselves can often form ‘codes’ which may be treated in the same analytical way as hand-coded data, however it is even possible to go beyond that. In section 3.3.10 we will explore a part of the system which uses computer vision techniques to auto-generate codes representing head nods and gestures, which can then be used in multimodal linguistic analysis in line with the work from that driver project. This was developed as a proof of concept to show what might be feasible in terms of automated coding.

#### 3.3.9 The DRS Document Viewer

A DRS Document (figure 3.23) is a special form of document which supports internal annotations. The purpose is to allow the user to construct a multi modal document containing not just text, but links to video clips and images. Further, the user is able to insert timestamps which when clicked jump directly to the related point in the analysis. Another feature is the ability to code a document, that is use DRS’s coding scheme to define set annotations about certain areas of the text. The DRS document viewer serves as a tool for the qualitative assembly of an interaction description document, one of the core tasks of an ethnographer.

Once a document has been coded, the user can use these codes as a special index. Down the right hand side is a list of the codewords, which when clicked, will open a new text file containing just the elements of the document which have been coded with that keyword. This serves as a quick way to get an overview of all the document’s contents related to a particular subject.

The document is actually an RTF file with some special data included at the end, so it can be viewed in other programs, however if edited this may break the coding structure.

#### 3.3.10 Gesture Tracker

As part of one of the driver projects - specifically focussing on the development of multi modal corpora, and in an attempt to mark-up the visual ‘mode’ of video data within DRS, we have used a computer vision technique to analyse videos from which gestures are recognised and annotated by the system in order to automate the creation of these coded annotation

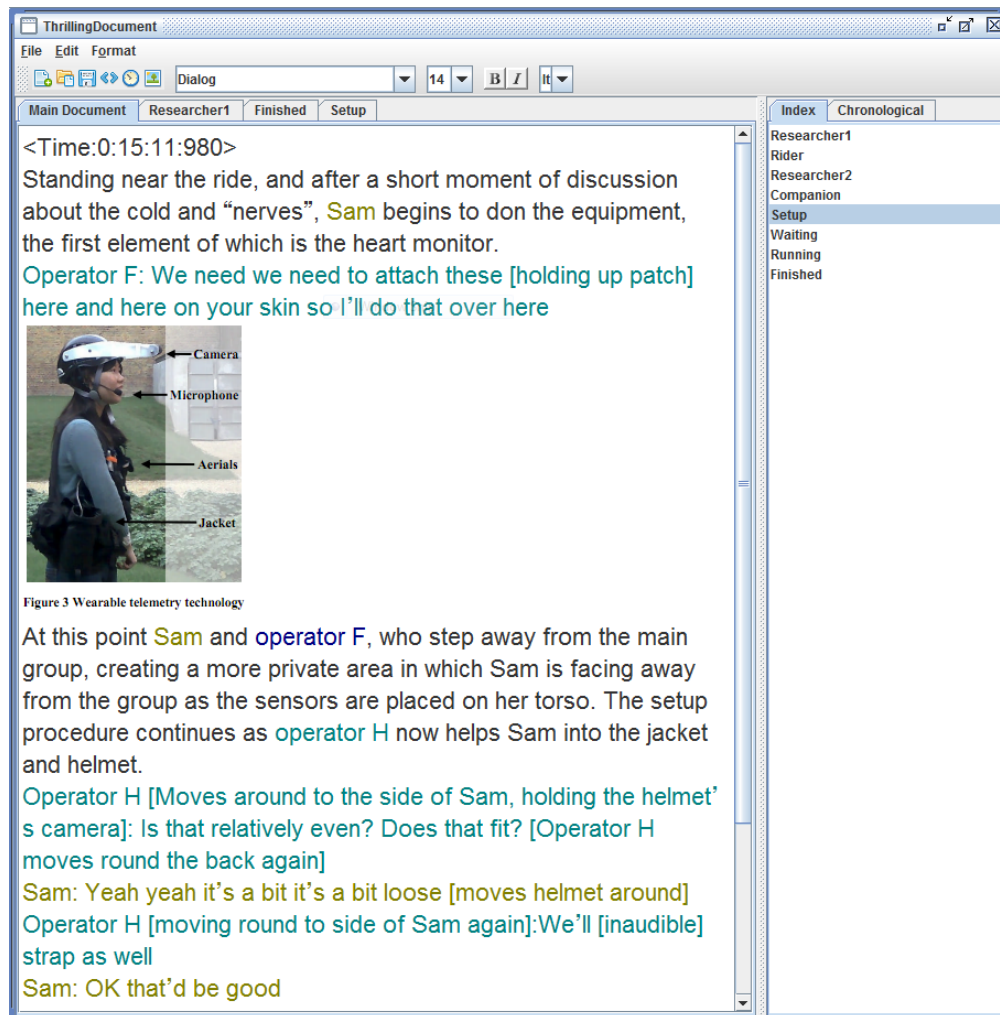


Figure 3.23: DRS’s Document Viewer.

tracks (this can later be encoded following rigorous analysis of the patterns between certain movement sequences). This is achieved by way of a tracking algorithm which can be applied to a video of a speaker and reports in each frame the position of, for example, the speaker's hands in relation to their torso, within a pre-defined granularity. Examples of the tracker in different modes are seen in figure 3.24.



Figure 3.24: DRS's Head and Hand Trackers(left and right respectively).

The tracking algorithm reports the position of, for example, the speaker's mouth in relation to their eyes, in each frame (for more information see [75]). The circular nodes seen in figure 3.24 (left) are the tracking targets (with a pre-defined granularity), which have the flexibility to allow the user to adjust the size of the tracked locations in relation to the specific size of, for example, the eyes and mouth of the participant (this has proved particularly useful when using close-up images in which participants have larger eyes and mouths). These targets are manually positioned at the start of the video and subsequently, as the tracking is initiated, a horizontal line is automatically drawn in the centre of these three nodes, marking an initial y-axis location (with position 0). Consequently, subsequent vertical head movements are denoted as causing a marked change in the y-axis in a + or ? direction (+ being a head up movement and ? being a head down movement).

The horizontal line also rotates to the left and right depending upon the position of the eyes, monitoring the angle of motion around the y-axis (when tracking head movements). The observation of the head angle from one tracked frame to the next proves invaluable to the analyst as such can help to reveal the characteristics of specific types of head movement (a feature that was integrated into the tracker as a result of various consultations with linguists), so for example such information can help to identify head shakes or head rotations as being distinct from a basic up-down head nod, should marked changes in the head angle be observed simultaneously with a marked change in the y-axis.

When tracking hand and body movement 3.24 (right), instead of the single horizontal line used as the point of movement reference in the head tracker, we are presented with three vertically positioned lines marking four zones on the image, R1 to R4 (R2 and R3 mark the area within shoulder width of the participant, acting as a perceived natural resting point for the arms, hence R1 and R4 mark regions beyond shoulder width). In consequence, the algorithm tracks the video denoting in which region the left hand (labelled as R by the tracker, since it is located to the right of the video image) and right hand (labelled as R by the tracker, since it is located to the right of the video image) are located in each frame. This provides an output of state for each frame. These states can then be applied to a video as a state periodic coding scheme.

The movement of each hand can therefore be denoted as a change in region location of the hand, so for example for Rhand (the left hand), we see a sequence of outputted zone 3 for frames 1 to 7, which changes to a sequence of zone 4 for frames 8 to 16. Ergo this notifies the analyst that the RHand has moved across one zone boundary to the right during these frames.

In theory, in order to track larger hand movements, the analyst can pre-determine a specific sequence of movements which can be searched and coded in the output data. So if, for example, the analyst had an interest in exploring a specific pattern of movement, considered to be of an iconic nature, i.e. a specific combination of the spontaneous hand movements which complement or somehow enhance the semantic information conveyed within a conversation, it would be possible to use the hand tracker to facilitate the definition of such gestures across the corpus (for in depth discussions on iconics and other forms of gesticulation, also see studies by Ekman and Friesen [70], Kendon [125–128], Argyle [97], McNeill [153, 154], Chalwa and Krauss [48], and Beattie and Shovelton [18]). Obviously the analyst would be required to train the tracking system by means of pre-defining the combination of movements to be coded as ‘iconic gesture 1’, for example (so perhaps a sequence of RHand or LHand movements into from R1 to R4 and back to R1 across x amounts of frames), in order to convert the raw output into data which is both more meaningful and usable.

Further to this, it is viable to note that in order to further enhance the efficacy of the hand tracker, the current prototype not only outputs the hand locations across individual frames, but also provides an ‘average’ location of each hand across the span of one second. Whereas the head tracker was designed to deal with the most subtle of head movements, some of which may last for less than one second, the hand tracker is designed to deal with

more emphatic, ‘large’ hand signals which may last 3,4 or 5 seconds, in addition to more subtle movements, as required.

Each potential gesture sequence (signified by the tracking output) is then labelled with a suitable code, and these codes are presented as a track of annotations anchored to the original video. These annotations may then be used in conjunction with the concordance tool (see section 3.3.11).

This automated approach to tracking hand gestures has two significant benefits over manual analysis. Firstly, there is the potential to save a great deal of time through automation. The tracker can be run much faster than many other image tracking approaches, working at close to real time. It is also possible for the tracker to produce a variety of different outputs at the same time. Secondly, tracking techniques should be able to more accurately recognise the intensity of gestural movement than a human observer can.

Mark-up specific gesture sequences without the tracker necessitates a more labour intensive approach, as we have seen in the previous section, similar to that taken by systems such as anvil and studiocode (as described in the related work section). A hierarchical ‘coding scheme must be defined in advance with necessary codes bound to particular keys ? then as a media file (typically, but not necessarily, a video) is played back, these keys can be depressed to signify an instance of a particular gesture code. These codes are then stored in a ‘coding track’ which can be exported to packages such as SPSS for statistical analysis, or used in conjunction with the text in the concordance viewer within DRS, allowing analysis of instances of co-occurrence between codes and utterances.

### **3.3.11 The Concordance Viewer**

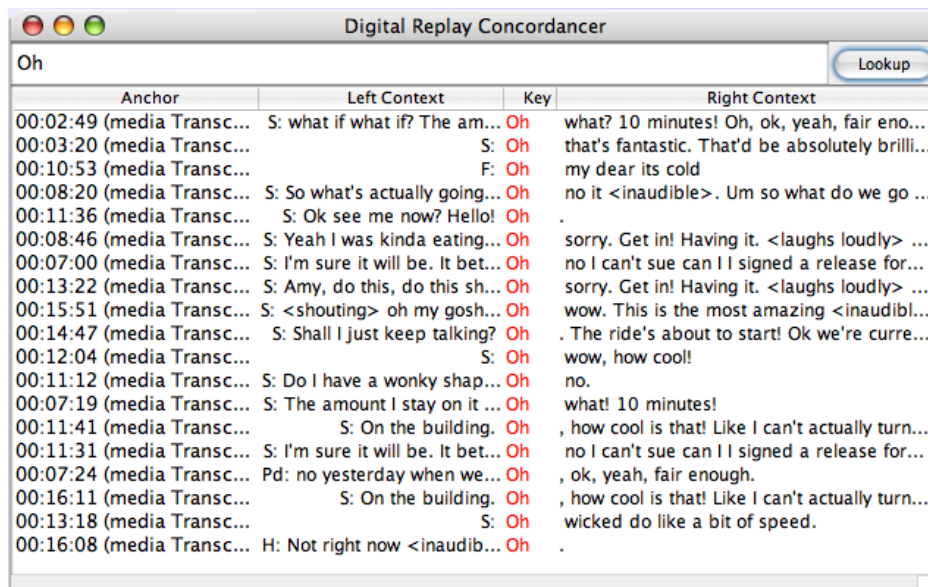
The concordance viewer (figure 3.25) is a special kind of search tool, developed as a result of the participation in the design process of members of Nottingham University’s English Linguistics department. It provides the analyst with the capacity for interrogating data constructed from textual transcriptions anchored to video or audio, and from coded annotations. The concordance viewer can utilise specific words, phrases, or other lexical tags in addition to any other codes, as a ‘search term’ (though this may also be a regular expression). Once presented with a list of occurrences, and their surrounding context (within the concordance viewer window - see figure 3.25), the analyst may jump directly to the temporal location of each occurrence within the video or audio clip. Further to this, the concordance viewer provides the commonplace frequency utility, giving raw counts of the frequency-of-use of



each search term, again providing an invaluable impetus for the quantitative exploration of data.

However, for many interaction researchers, it is the complex interactions between dialogue, non-verbal behaviour and system activity that are of most interest. There has been an increasing interest in the relationship between certain nonverbal behaviours and the linguistic context in which they occur as researchers recognise that they can be tightly synchronised in natural language (e.g. to convey shared meanings, [17]) The study of these relationships will lead to a greater understanding of the characteristics of verbal and non-verbal behaviour in natural conversation and the specific context of learning, and will allow social scientists to explore in more detail the relationships between linguistic form and function in discourse, and how different, complex facets of meaning in discourse are constructed through the interplay of text, gesture and prosody (building on the work of McNeill [153] and Kendon [126,127]).

Since the concordance tool treats textual and coded annotations in the same way internally, the search functionality extends to all media within an analysis. This allows an investigation of all types of annotation in the same manner, applying the same skills and techniques used in the analysis of traditional corpora (for an example see [192]). This multimodal concordance viewer has led to the need for developing new approaches for coding and tagging language data in order to align textual, video and audio data streams [6,132].



Anchor	Left Context	Key	Right Context
00:02:49 (media Transc...	S: what if what if? The am...	Oh	what? 10 minutes! Oh, ok, yeah, fair eno...
00:03:20 (media Transc...	S: Oh	Oh	that's fantastic. That'd be absolutely brilli...
00:10:53 (media Transc...	F: Oh	Oh	my dear its cold
00:08:20 (media Transc...	S: So what's actually going...	Oh	no it <inaudible>. Um so what do we go ...
00:11:36 (media Transc...	S: Ok see me now? Hello!	Oh	.
00:08:46 (media Transc...	S: Yeah I was kinda eating...	Oh	sorry. Get in! Having it. <laughs loudly> ...
00:07:00 (media Transc...	S: I'm sure it will be. It bet...	Oh	no I can't sue can I I signed a release for...
00:13:22 (media Transc...	S: Amy, do this, do this sh...	Oh	sorry. Get in! Having it. <laughs loudly> ...
00:15:51 (media Transc...	S: <shouting> oh my gosh...	Oh	wow. This is the most amazing <inaudibl...
00:14:47 (media Transc...	S: Shall I just keep talking?	Oh	. The ride's about to start! Ok we're curre...
00:12:04 (media Transc...	S: Oh	Oh	wow, how cool!
00:11:12 (media Transc...	S: Do I have a wonky shap...	Oh	no.
00:07:19 (media Transc...	S: The amount I stay on it ...	Oh	what! 10 minutes!
00:11:41 (media Transc...	S: On the building.	Oh	, how cool is that! Like I can't actually turn...
00:11:31 (media Transc...	S: I'm sure it will be. It bet...	Oh	no I can't sue can I I signed a release for...
00:07:24 (media Transc...	Pd: no yesterday when we...	Oh	, ok, yeah, fair enough.
00:16:11 (media Transc...	S: On the building.	Oh	, how cool is that! Like I can't actually turn...
00:13:18 (media Transc...	S: Oh	Oh	wicked do like a bit of speed.
00:16:08 (media Transc...	H: Not right now <inaudib...	Oh	.

Figure 3.25: DRS's concordance viewer.

In figure 3.25, the concordance viewer shows instances of a search term ('Oh') occurring

within all annotations associated with the project (transcriptions, codes and free annotations). The context in which that term occurred is shown to the left and right. Selecting an instance will jump DRS to that analysis and location, automatically opening the media that the annotation pertains to at the correct time.

Clicking on any one of the cells offers the ability to jump to the time in the analysis in which that instance occurs. Unlike other viewers the concordance viewer searches all the data in a given project, rather than just within an analysis. In this case the data may be taken from many non temporally related media, therefore keeping them in a single analysis may not be appropriate, however the need exists to search across each of them. In the cases where a token is not in the current analysis, DRS will switch to the correct analysis in order to display the appropriate data.

### 3.3.12 Log File Workbench

One of the core features of DRS is of course the ability to deal with so-called system log data. DRS provides a tool called the Log File Workbench examined in more depth in chapter 4) with which to import and create views over that data. The design and functionality of the log file workbench will be discussed in the next chapter. For now it is mentioned only to make the reader aware that DRS *does* support synchronized playback of system log data, indeed DRS provides some basic charts similar to those developed for Replayer (time series, event series, histogram etc).

### 3.3.13 Summary

DRS provides a series of tools for viewing and marking up data in a number of different ways. Much attention is paid to creating usable tools for annotation and coding (automated coding in the case of the computer vision system) - in particular working with the social scientists who are using the system in anger, to iteratively prototype those tools and provide exactly the necessary facilities to conduct interesting research, for example the concordance tool directly supports a regular practice from corpus linguistics in a way that a basic search tool simply could not. While DRS focusses on the markup and description of qualitative data, it should not be forgotten that turning system logs into qualitative data is a necessary step, which once achieved gives it the same significance as other marked up data - and thus it can be approached in the same way and with the same tools.

## 3.4 Conclusions

We have now seen much of the functionality of both systems, and the very different focusses should be clear. As we will see in the next chapter however the systems are really not in any kind of conflict. In fact in general the weak points of one tend to be the strong points of the other. The stronger influence of social scientists in the design of DRS lead it to feel more like a ‘familiar’ CAQDAS tool, and it provides support for many of the kinds of methods used in qualitative data analysis. Its weaker support for generalized analysis of log data reflects the priorities of those social scientists, as well as their lack of experience in handling log data. Replayer on the other hand, coming as it does from a more directly computer science oriented background has strong support for handling the logs rooted firmly in the field of information visualization, but lacks the crucial support for methodological synchronization that would make it a tool more generally usable in qualitative social science.

## Chapter 4

# Key Implementation Factors

### 4.1 Introduction

This chapter will focus on three specific areas of the two systems described in the previous chapter (Replayer and Digital Replay System). Those three areas are:

- Log file handling
- Distributed software architecture
- Synchronization

In each case we will explore how the systems handle this area, how they differ, and what the benefits and drawbacks of each approach are. It is the handling of these three areas in particular which make Replayer and DRS distinct from many of the computer aided qualitative data analysis tools discussed in the related work chapter.

### 4.2 Log Files

#### 4.2.1 Introduction

The feature that really sets Replayer and DRS apart from previous systems is the fact that they are designed to handle system log files - and do it in a way that allows them to be effectively combined with other heterogeneous media. Both systems have components designed to read in log files and store them in an internal database. Again, the systems differ somewhat in approach.

### 4.2.2 State and Event Logging

Replayer in particular relies on an idea whereby system logs can be separated into two distinct types:

- State Type
- Event Type

State type logs are characterised by being representable within a single consistent database table. They essentially come in two forms *regular* and *irregular*. We can consider state data as data that is sampled, either at regular intervals, or irregular. The key point however, is that it is the type of data that can be used to reconstruct the *state* of a system at any given time. State data is well suited for display on certain types of graphs such as time series. The fact that state data can be stored in a single table, makes handling it relatively simple for viewers. Conversely we have the event type data. Event data typically describes irregular discrete occurrences. While in some cases it may be suitable to describe this data as irregular state data, the size of the table necessarily increases with the number and complexity of events. This method is more suitable for recording things like user interface interactions. So as a simple example, Imagine a thermometer application. It runs constantly, but the user must press a button to check the temperature. The temperature is regularly sampled and stored as state data, however we might store the user's checking of the values as event data. Assuming we sample every 30 seconds, we have a regular state value, that can be plotted easily on a time series. We can also plot the user's interactions with the system on an event series. Selecting a user interaction event in the event series we will be able to look up the last sampled 'state' at that particular time. To achieve this with a single table would require the insertion of a potentially irregular row in the state table and necessitate an extra column(s) to describe the user's interaction. This simple separation of logging types leads to an extensible generalised architecture for storing logged data in a database. With that standardised system we can significantly reduce the complexity required for viewer components to be able to handle the data. It is sufficiently flexible to handle virtually any type of system log.

### 4.2.3 Replayer's Log File Handling

#### 4.2.4 Log Files and Parsers

When considering the types of log file Replayer would have to handle it became immediately clear that this was no small task. Around this time there was a seminar within the Equator Interdisciplinary Research Council where an attempt was made, driven primarily by researchers from the University of Southampton who had been working on developing semantic web applications, to create an extensible format for system logs that would be applicable to any system. This was an XML (Extensible Markup Language) and RDF (Resource Description Framework) ontology based approach [201]. While having a standardised logging format certainly offered plenty of advantages for builders of replay tools, this also presented two major disadvantages. First the framework was extremely complicated - necessitating serious work for anyone wishing to write to that standard, and secondly that sticking exclusively to this format would preclude any system from examining legacy data without first changing the logs into this new format, or building legacy-parsers - something that would have to be approached by a programmer familiar with both the new standard and whatever the present standard of any given log might be. Instead it was decided to put that standard aside temporarily, given it had not been fully agreed at the time - and indeed was never fully agreed upon or implemented in the lifetime of the Equator project, and create a simpler standard which could easily be used locally, while providing facility to parse legacy or alternative logging standards, including that one when it was complete.

To get the data from a system log into the database, and thus make it queryable and subsequently visualisable, each log type would require to be parsed, that is, examined programmatically. A decision was taken to make two parsers - one based on a simplified version of the above standard, called, for want of a more imaginative name, Replayer Markup Language (RML), and another to analyse simple text logs, which would serve as a template for the development of legacy parsers. Local developers would be encouraged to record their logs in RML which provided facility for recording both the state type and event type logging discussed above.

Here we can see a sample of and RML log:

```
<record type="event">
<timestamp>632597507190000000</timestamp>
<HRTIME>Tue Aug 16 01:58:39 BST 2005</HRTIME>
```

```

<apspotted><mac>00032F178046</mac>
<ssid>MalcNet</ssid>
<type>0</type>
<crops>0</crops>
<fruit>apple</fruit>
<picked>0</picked>
<version>0</version>
<reseed>False</reseed>
<wants>apple;apple;apple;apple;apple</wants>
</apspotted>
</record>

<record type="event">
<timestamp>632597507220000000</timestamp>
<HRTIME>Tue Aug 16 01:58:42 BST 2005</HRTIME>
<peerfound>
<id>350C1C011B0604F11800-0050BF1977E0</id>
<two>null</two>
<three>null</three>
<zero>null</zero>
<one>null</one>
<four>null</four>
<user>Alex</user>
<scanstop>null</scanstop>
</peerfound>
</record>

```

So to get the data from a log file into the database, the appropriate parser component is opened and pointed at the appropriate file and the parsing process is started. It then automatically connects to a running server as all clients do (of which more in the next section). This parser generates a set of database creation instructions which are then executed by the server. As an interesting by-product of using this technique, it became possible, though this technique was never actually applied, for a connected device to write its logs directly into Replayer's database, thus completely bypassing the need for parsers and making such logs accessible for analysis in pseudo-real-time.

#### 4.2.5 Instrumentation

With the RML specification in place, it became apparent that Replayer would have to provide a way to encourage developers to record their logs using this standard. The chosen approach was to develop an instrumentation tool which would automatically add logging code into a program from within the IDE in which it was being developed. Systems like GRUMPS [152] adopt a dynamic proxying approach to insert logging code directly into java

bytecode making the logging entirely transparent. However, Replayer's *Instrumentor* component takes a different approach. Logging code is instead inserted visually into the source code (figure 4.1). While dynamic proxying aims to make the instrumentation transparent to the programmer, direct code insertion takes the opposite approach and allows the programmer to see, and indeed modify the code inserted by the instrumenting system. Indeed, the Instrumentor actively encourages, and even requires the programmer to interact with the logging code. The reasoning behind this opacity is the intended use. While GRUMPS appears to be intended for use by a third party, Replayer's Instrumentor is designed to be used by the designer/coder of a project, thus a user will have intimate knowledge of the code.

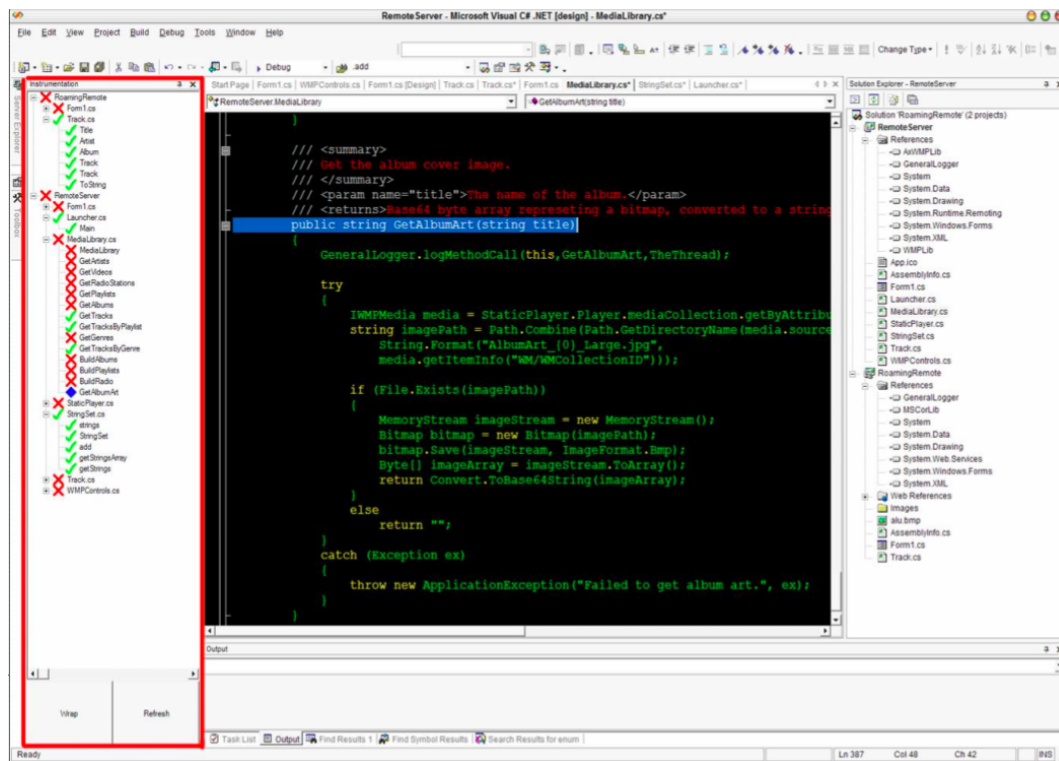


Figure 4.1: Replayer's instrumentor running in visual studio (highlighted in red). Ticked classes/methods have been marked for logging.

By allowing designers to adapt the instrumentation, they should be able to control, to an extent, the performance hit caused by that instrumentation. This is taken a step further in the case of variable logging: The user has to actively decide the best time to call a variable dump method, ensuring that it is called at the most suitable time. Another advantage of this transparent method of instrumentation is that the designer can create meaningful output strings, that is, the additional text messages output by the logger system for a particular



method. This means that when visualising the data in Replayer, the user will have a clear schema of what individual log atoms refer to. Indeed it could thus be tailored in such a way that the logs could be effectively explored by a user with less knowledge of the code. One side effect of using the instrumentor is the additional workload generated, and the effects of code bloat [209], however if a system *is* to be logged for later analysis, then Replayer's instrumentor provides a simple approach to including logging code in a project, and the visibility of the logging code mean steps can be taken to manage code bloat. Replayer's Instrumentor is an add-in for Microsoft's Visual Studio 2005, designed to simplify the logging process (figure 4.1). A tree view shows all the classes, methods and variables in a solution with icons showing whether or not they are currently being instrumented. A user can apply or remove instrumentation at any level of the tree allowing the granularity (and subsequent performance hit) of the instrumentation to be tailored to the user's requirements. Visually the tree structure and icons recall CVS and other code management systems, with checked classes being logged and crossed ones ignored. This version of the instrumentor is of course only applicable to programs written in C#, however the same principle could easily be applied in for example a plugin to *Eclipse*, or any other IDE which supports third party plugins.

#### 4.2.6 Log File Parsers

In order to begin the process of analysing log files, it is first necessary to include the contents of those files in the Replayer database. To this end, some special clients are required to parse those logs and generate the Standard Query Language (SQL) commands needed to build the database. Replayer makes no initial assumptions about what will be contained in these logs, but works on the premise that some data can be considered as state-type, and some as event-type. Such logs may not be separated in the way that those of treasure were, indeed many logs consist only of event data and as such state data is only implied and must be inferred from the events.

Replayer provides two parsers by default - one to parse logs in the RML format, and another template parser, which contains a set of methods designed to simplify the process of creating a custom parser. Because the server makes so few assumptions about the content of the logs, it is the job of a log file parser to create and populate the database based on that content. The actual implementation is up to the user, but the template includes methods for reading in files, generating SQL *CREATE* and *INSERT* commands, and for sending those

commands to the server.

### RML Log File Parser

The RML Based logfile parser is able to parse any system log created using the RML standard. This is an XML based standard and as such the parser uses Java's SAX parser to examine them. As the logs are parsed in a serial fashion this is the most appropriate. An Entry in an RML log looks like this:

```
<record type="<type>">
<timestamp>value</timestamp>
<eventname>
<variablename 1>value</variablename 1>
...
<variablename n>value</variablename n>
</eventname>
</record>
```

where attribute 'type' is either 'state' or 'event'. Note that in the case of state logs, the 'eventname' tags are excluded.

The state table is generated based on the format of the first log entry to contain the state attribute. To do this, an SQL *CREATE TABLE* command is sent to the server, which automatically modifies it to include RID as a field. This, and every subsequent entry with the state attribute then generates an SQL *INSERT INTO* statement, again sent to the server to populate the state table. Similarly events are handled in much the same way. Each event generates an *INSERT INTO* for the event master table (in place by default with a new Replayer database) and a *CREATE TABLE* command to create a suitable table to contain the variables for this event. The server then ignores the duplicate create table events. Using this approach, while somewhat wasteful of traffic means that new events can be added at any time, as and when they are encountered in the log files. Also different parsers can be used for different log formats within a single session, and because the server ignores requests to create tables that already exist, there is no error thrown when a second parser tries to create a table that already exists. Once the appropriate tables have been created, it is simply a matter of generating more *INSERT INTO* commands to fully populate the database.

### CSV Parser

One common format of system logs is that of comma separated values. Thus we created a special log file parser for Replayer that handled these files, allowing sample data from HiVE [180] to be examined with Replayer. It works in a similar fashion to the RML parser, though the CSV format is perhaps less versatile than that of RML, as it is only really suitable for state based data, with the table headers being defined in the first line, their types in the second, and values in all the proceeding lines. Indeed, Replayer itself uses this particular CSV format to pass its internal data around between clients.

### Custom Parsers

It is of course simply not the case that all future log files will be written in RML or CSV. Indeed, even within a local research group that knew of and expected to use Replayer, this was frequently not the case, and of course there remained the question of legacy data. As such it was clear that additional parsers would be required. After writing a number of these, the logical solution seemed to be to crate a template and library which would allow users to easily create their own. The template sets up the necessary communications with the server, and reads in a user-selected file. It then offers a library of methods for generating SQL statements, one for table creation and another for data insertion. With a little work going into the control statements to actually read the contents of the file, something it is not easy to offer help with, without knowing what format that data will take, it is thus possible to create a parser which will create and populate the server's database with appropriate data.

#### 4.2.7 Log File Handling in DRS

DRS handles log files somewhat differently. Unlike Replayer it doesn't specify its own format instead relying on an extensible system of text file processors to handle logs in a variety of common formats. It also doesn't specify a way in which its databases should be organised. While in theory this is more flexible, in practice it means that the task of constructing generic data viewers for system log data becomes somewhat more complex.

### The Log File Workbench

In DRS log files are imported, turned into databases and viewers are configured for those databases using a system called the log file workbench. This system allows any number of databases to be created, which are then treated as ‘media’ objects by the rest of the DRS system. Figure 4.2 The log file workbench uses a series of *processors* to handle the importing

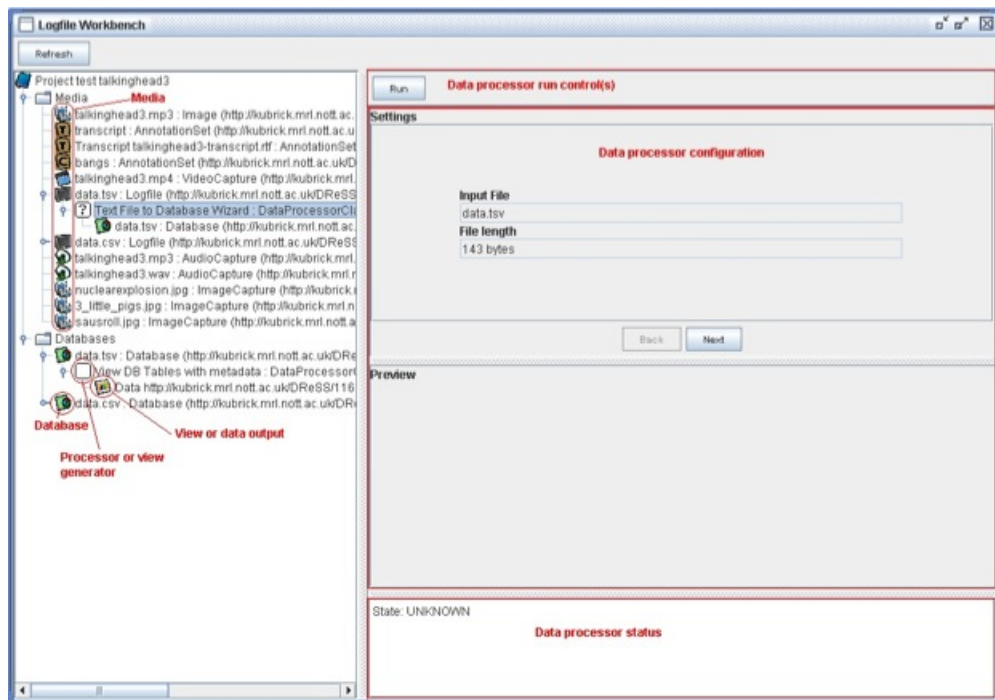


Figure 4.2: DRS’s LogFile WorkBench

of logs. There is one configurable processor designed to handle generic file types such as comma Separated value files (CSV) or tab separated value files (TSV). The processor takes the form of a *wizard* (shown on the middle right in figure 4.2) in which the user specifies information about the data contained in the file, including encoding, separators, headers, footers and meta data about the type of data contained in the columns. This processor is significantly more flexible than Replayer’s CSV importer. Once configured, the processor is run over a text file producing a single table database which then appears in the list of project media (shown on the left in figure 4.2). For state-type data this is an extremely effective system. However, when handling data in more complex forms than simple tabular data the task becomes more challenging. The log file workbench provides an API for creating new processors which can be used to create any style of tables necessary. This means that in theory any logged data can be imported to the system, but like Replayer it will require a

specific processor to be written for any complex type of log. Unlike Replayer, DRS is able to support multiple databases, which in theory means that several different datasets can be viewed simultaneously.

### **Log File Meta Data**

Unlike Replayer which uses a rigorously defined database table structure, DRS allows tables to be in any form. To make it possible for viewing components to work on those tables in a generic way, DRS allows the user to apply meta data to particular columns, defining them for example as timestamps, strings, integers, or filenames. This more flexible approach allows timestamps for example to be in virtually any format. As long as that format is defined in the meta data then DRS's internal time management tools are able to turn it into a usable form. While this increases the difficulty of actually inputting logged data to the system, it does allow the potential for meta data based searches across the databases, and because the metadata ontology is extensible, the volume of meta data is limited only by the particular processor, and the amount of time the user is willing to spend including it.

### **Creating Views**

Once the log file has been imported a secondary processor can be applied to the created database to construct views on the data (created views shown on the lower left of figure 4.2). Again there is a generic processor available, which is simply a Synchronized table viewer, selecting the cells associated with the time currently selected in one's DRS analysis (assuming there is a timestamp column in the database). Additional views can be created by anyone with some java programming experience by implementing the viewer interface.

#### **4.2.8 Comparing the two approaches**

Once again we have two systems doing a similar thing in somewhat different manners. If a program is instrumented to output its data in RML form, then importing log file data is significantly easier with Replayer, however DRS provides far more flexibility for importing data in a series of standard forms. DRS also benefits from being able to maintain several databases. In general, when it comes to importing data, DRS is the more sophisticated of the two systems, however once the data is in there, DRS offers very little in the way of things to do with your data, without writing custom viewers, such as the one we will see in the case study in the next chapter. Replayer, on the other hand has a strong selection

of viewers of various types, detailed in the previous chapter. These generic viewers, while not providing the specific functionality of a bespoke tool, provide a variety of methods for visually querying and exploring your data. DRS as a single tool, has a greater sense of internal cohesion, with the same metadata applicable to database tables and columns as it is to transcripts, media files or code atoms, however, because it lacks Replayer's state/event separation, at least by default, as well as missing out on Replayer's RIDs it is limited to synchronising data by time alone. From an extensibility point of view DRS requires its processors to be written in java and requires a specific interface to be implemented for its log file processors and viewers. It does however, provide all the necessary methods and tools necessary to quickly create processors. Conversely, Replayer, while it provides a template parser which does contain the necessary tools, has no such restrictions. Parsers, like other viewers, are essentially separate programs and can thus be written in any language as long as they can send messages via TCP to the network server using the Replayer communications protocol. Despite this flexibility it would appear DRS has the superior engineering for reading in log files. The configurable handling of so many potential file types is far superior to that of Replayer, but Replayer's rigorous database structure and comprehensive set of synchronized generic data viewers make it significantly more useful once the data has been input. It is perhaps unsurprising that DRS is having its database system rewritten into a structure resembling Replayer's. However, it will maintain its superior support for input, and metadata, as well as the support for multiple databases crucially missing from Replayer. DRS is also having a set of generic graph type viewers similar to those of Replayer written to allow the same kind of coordinated data exploration that Replayer currently offers. This is an example of the convergence of the two approaches.

## 4.3 Distributed software architecture

### 4.3.1 Introduction

Replayer and DRS both have a client server architecture, to allow multiple users to work on an analysis, but they are approached in very different ways. Replayer is created as a set of networked viewer components which all communicate with a workgroup database and thus allow users to work concurrently on the same data, or a single user to utilize two or more computers to circumvent physical limitations of screen real-estate and processing capacity. DRS on the other hand, uses an approach more like concurrent versioning system (CVS),

with each client self contained and a networked server allowing non-simultaneous shared access to projects and files stored elsewhere. In practice we see that while the potential collaborative use of Replayer was not fully exploited in our pilot studies, the architecture did nevertheless allow for a more customised setup. DRS, while somewhat simpler with respect to collaboration, is the more heavyweight of the two programs with many users noting that it had a very difficult learning curve.

### 4.3.2 Replayer's Client-Server architecture

One of the design goals Replayer has been to allow for networked interaction, with several computers sharing a single Replayer session and subsequently support both collaborative analysis, and distributed task sharing. The design would therefore require a central session controller and clients to that controller, thus necessitating the classical client-server architecture. It was short logical step to assume that that session controller would serve to control access to the database. Each client is an independent program which communicates directly with the server using simple network protocols, thus allowing clients to be run on any computer sharing a network link with the server. This allows differing configurations based on the particular requirements for any given session. The server broadcasts its existence across the network on a known port, and each client, when started, listens on that port, locates the server and sets up a connection. By using a simple communications API, it is possible to allow clients to be written in any language, by anybody with some programming experience, and no one client need depend on the existence of any other. Additional components can be added to the set at any time, including runtime.

This architecture has two distinct advantages. First, two or more analysts are able to simultaneously collaborate on the same piece of work simply by connecting their computers to the same network server, and second it is possible to use distribution to share out the workload between multiple computers. Let us take an example in which we have a number of videos which must be played simultaneously. Now add to this the requirement that they must all be high resolution and encoded with high visual and audio detail. Add to that the need to process some very complex log files, and possibly visualise them with a CPU intensive dynamic graphing technique.

One way of encoding these videos might be MPEG2 (the DVD standard). Playback of MPEG2 is quite processor intensive - an examination of playback of an MPEG2 movie at television resolution (720x576 pixels) using a computer with a reasonable processor (2.4Ghz

Pentium 4) and plenty of Memory (1Gb) using Cyberlink's PowerDVD <sup>1</sup> software showed the playback process consistently using 75% of the CPU. Trying to play two of these movies simultaneously becomes problematical. Add to that the system log data analysis and it quickly becomes impractical. Instead consider the situation where we have several computers available. We can dedicate one each to the two movies and a separate one to the graphing and have everything run smoothly. Of course this necessitates having several computers, and is therefore not an option for every user, however it does make this kind of operation possible. One other side benefit comes from using multiple computers. It quickly became apparent with use that Replayer is extremely hungry for screen space (sometimes called screen real estate). At the time of development - A good high resolution monitor screen was typically 1600x1200 pixels, with 1280x1024 being more common. Even with two monitors it wouldn't take many videos and visualisations to completely fill those screens. Figure 4.3 is a photograph of Replayer running simultaneously on two computers and four screens) There are in fact only five different coordinated viewers showing. The views are (clockwise from the top left) a pair of videos, an event series, a time series, a histogram, a Google earth bridge (and associated Google earth view). Note also that the laptop on the left is running Apple's OSX, while the laptop on the right runs Microsoft's Windows XP. The screens shown here can be seen in closer detail in figure 3.2.



Figure 4.3: Multiple networked instances of Replayer

---

<sup>1</sup>[www.cyberlink.com](http://www.cyberlink.com)



### 4.3.3 The Server

Replayer's server, called the *Control Unit*, holds the database and manages connections to a number of data viewers as clients. The database itself could have been implemented as one central webservice based database (a grid-style [79] approach), that all clients of Replayer connect to allowing for global sharing of data, or each client could have maintained its own database, synchronising with other connected clients when necessary. For a proof of concept system, a one server per session approach was used based on a hypersonic SQL database<sup>2</sup> and the QuickServer java networking API<sup>3</sup>. One computer would maintain the database only for the duration of a given Replayer session. Other clients could connect during that session but they would interact with *that* dataset rather than whatever replayer databases they may have stored on their own machines. To further simplify the cross system design, it was decided that each viewer would actually be a separate program, and communicate through network protocols with the server.

The server broadcasts on a user defined port a simple message containing the IP address of the computer on which it is running and on which port connection requests should be made. Once a client connects, the server sets up a bi-directional TCP link over which all communications with that client will occur. Only the server uses the Quickserver package, with clients using a simpler stream writing technique. This allows, the cross-language support mentioned earlier, as dependency on quickserver would limit the clients to those programmed in Java.

### 4.3.4 Communications Protocol

Replayer uses TCP for all its internal communications based around a very simple structure. Messages are transferred in uncompressed ASCII, over the QuickServer defined system using keywords, which the client and server recognise and use to trigger the appropriate responses. Keywords are denoted by the '\$' character and listed in the API. An example might be a message sent to the server requiring a response containing data in the database to be sent to a particular client. This would take the following form:

```
$SQL<sql query> $CLIENT <client ID>
```

The \$SQL tells the server that this is going to be a selection query, it then sends that query to the database. The response from the database is the routed to the appropriate

---

<sup>2</sup><http://hsqldb.org>

<sup>3</sup><http://www.quickserver.org>

client, denoted by the number following the \$CLIENT keyword. Note that this client may not be the same one that sent the request - indeed requests are typically sent from the *Meta Tool*, with their responses being sent to one of the visualisation clients. The server maintains a list of active clients, each with a unique ID so the server is able to use this list to decide how best to route the response. There are also cases, primarily those of selection, where the server will multicast a response to all its clients, which it does simply by iterating through that list. A kill protocol is used when individual clients are closed to ensure that the client list is accurate, and regular ping-type communications are made to ensure that each client is alive - this means that should an individual component fail, the server will be aware of this and remove it from its list.

This protocol, while very simple, allows the development of additional clients with relative ease, not requiring complex communications systems, though it is not appropriate for sensitive data as the communications are unencrypted. However, were stronger security a requirement, Quickserver offers the facility to include encryption on all messages. This would of course add an extra layer of complexity to constructing new client visualizations.

### 4.3.5 The Database

The back end database, is created with the hypersonic SQL package, and designed to run in main memory, until such time as the size of the database exceeds the available memory - defined when the virtual machine is started, at which point it switches to a more traditional format. The database is written to XML on shutdown, and recreated from that file on startup. This allows for several databases to be maintained simultaneously with the user switching between them as required. As such, datasets are typically stored in separate databases, with only one being active at any given time. Importing and exporting is then simply a matter of sharing the XML files, though they are packaged up into jar files to achieve this as one database may generate more than one file. The server uses JDBC to communicate with the database system allowing for the execution of SQL commands. As with many database systems, there is no direct visual interface to the database, though it is possible to interrogate it using freely available database viewer tools included with the hypersonic SQL package.

### Database Architecture

As we have seen in the Log file handling section, Replayer's database makes a differentiation between the idea of state and event type entries to the database. The defining factor of these paradigms is that state will be a set of values (not necessarily continuous), but that these values will be all of the same types, and thus easily represented with a single table. The structure of that table is defined by the log file parser, but the first two fields must be included. First, *Replayer Identifier (RID)*: this is a unique ID for that row, used as the primary key of the table and also used by the selection process to achieve brushing and linking. Because the RID must be unique it is not possible to expect a log file parser to be able to assign this; several different independent log file parsers may be used to create one database, or just one used several times. Because this could happen concurrently it made more sense to allow the server to assign this crucial attribute. Thus, it is assigned by the server, which modifies incoming SQL Create or INSERT commands to include this field, and assigns them based on its own internal list. The second value is always *TIMESTAMP*. This is self explanatory, but Replayer expects it in the form of Milliseconds since January the first 1970 - which is the standard that Java uses for creating time values. There are of course many different ways in which time can be represented, but Replayer makes it the responsibility of the log-file parser to correctly translate these into a Replayer-friendly format. All the other fields in the table will be defined, again, by the log file parser. The first command it is expected to execute is an SQL CREATE TABLE for the state table. A simple example of a very simple state table might include:

RID, TIMESTAMP, XPOSITION, YPOSITION

This particular table thus includes the RID and timestamp as required, but also the x and y positions of a user. The exact form of these coordinates, be they British Ordinance Survey, Latitude/Londitude, or some form of relative positioning is not known by the database, and it is up to the clients to correctly interpret them. The state table is thus the simplest of database tables, and easily represented. Slightly more complex is the case of events. An event may have a number of values associated with it, but those values may differ from an event of another type, therefore a single table is inappropriate. Instead Replayer creates an event master table, containing just three values for each event - *RID*, *TIMESTAMP*, *EVENT TYPE*. Each event type refers to a second table, specific to that event, using the RID as a foreign key and is created with whatever fields are appropriate for that event. It

is thus possible to modify the database's set of tables at runtime to include more tables for different event types as they appear in the process of parsing the logs. Let us consider the case of an event that records a button press on the UI of a recoded system. The event master table would include a row containing an *RID*, timestamp, and the event type, in this case *BUTTONPRESS*. The server, on receiving this update will examine the list of tables to see if *BUTTONPRESS* is a valid event type. If it is, then the vales are simply written into that table, if not a new table is created, with fields based on the content of the event. That table might look something like the following example:

*RID, BUTTONNAME*

It is of course likely that an event may have far more variables associated with it, but let us just consider this very simple example for now. If we imagine that we have the three tables described above, giving us the State table, The event master table and the button press table, we can work out for any given event - when it happened and where the user was at that time. This allows us to create, for example, spatial or temporal distributions of button press events. One more table is required to complete this set. Replayer supports not just logged data but also the use of multimedia data types. The most common of these are video and audio, and in order to accommodate their Synchronization within the system, they too must be included in the database. One common feature of such files is a tendency to be temporal - that is they have a duration. Note that this is not the case with still images but these are defined as having an instantaneous duration so can be represented in the same way as other media types, and at the time of writing Replayer does not have an image viewer component. The media table thus contains *RID, START TIME, END TIME, FILEPATH, TYPE* values, with type typically containing the codec with which it is compressed - which also defines what type of media file it is. With media files, Replayer does not support streaming, instead requiring the file to be stored on the local computer.

#### 4.3.6 Queries

When Replayer receives a query from a client it executes the appropriate SQL command to the database, and the result comes back as a java result object. One option would be to serialise this object and then route it to the appropriate client, however because of the cross-language philosophy underlying Replayer's component architecture, it instead converts that result into a comma separated value table. The first line of this table contains the field

names, the second, the field types, then each subsequent line contains the values. An example response may look something like this:

```
RID,TIMESTAMP,XPOSITION,YPOSITION
INT, LONG, DOUBLE, DOUBLE
1,1000001,12.643878,65.87459847
```

using newline characters to separate the lines. It is entirely up to the client component how that data should be represented, indeed it is by appropriate choice of visualization components that researchers are able to extract accountable data from the system logs stored in the database.

### 4.3.7 Clients

In the Replayer architecture (figure 4.4), everything except the control unit, is considered a client. This can include, but is not limited to, log file parsers, visualisation components, multimedia players and bridges to third party programs. It is also possible for a networked system that is being captured to be itself a Replayer client and thus write its logs directly into the Replayer database. Each client conforms to the Replayer communications API and fulfils the necessary requirements to be loaded by the meta tool as outlined in the section on extensibility (section 4.3.9). In the proceeding subsections we will examine these different types of component and consider how they work and integrate with Replayer as a whole.

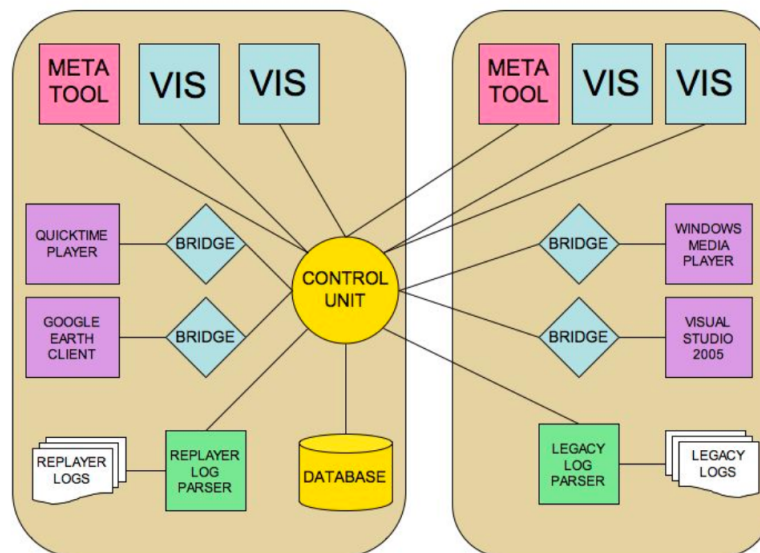


Figure 4.4: Replayer's System Architecture

### Visualisation Components

While the set of visualisation components are quite diverse in style, they do all follow a consistent formula.

- Start up empty and register with the server
- Wait for some data from the server (sent via a request from the meta tool by the user)
- Filter the data in some way
- Display the data
- Allow the user to make selections
- Reflect the selections made in other components.

Communication with the server is achieved over a TCP connection mediated by the server (figure 4.4). A simple API of available message types offers the client various options for this communication. Assuming it is written in java, a client must implement the *iReplayerClient* interface which requires the client to include *connect()*, *recievedata()*, *sendselection()*, and *recieveselection()* methods. This, when combined with the *ReplayerClientNetworking.jar* library provides everything the client needs to communicate with the server. The *Replayer-ClientNetworking* Library is fairly simple, and replicable in any language that supports raw SOCKETS, so the dependency on java is not complete, but it is certainly simplest to create one's clients in java.

Data are received from the client in the form of a comma separated value string using newline characters as line separators. It is entirely up to the client the way in which these data are displayed to the user. Descriptions of each of the default set of components is included in the previous chapter.

When a user makes a selection within a visualisation component, the component must send to the server a list of all the selected RIDs. In the cases where the user is selecting a slice of time, the component must find all the RIDs within its stored data, that occur within that timeslice, and send them to the server. The server will then respond with a list of selected RIDs, multicast to all the connected clients.

When a client receives a list of selected RIDs from the server it must update its selection to include only that set of objects. In some cases, such as that of the event series, this is

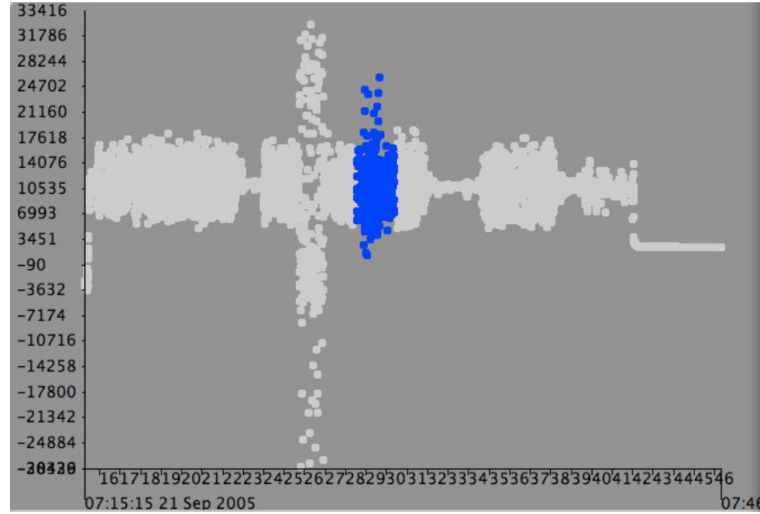


Figure 4.5: A partially selected event series.

trivial - simply graying out those glyphs that are not in that new set of objects (as shown in figure 4.5).

In a more complicated viewer such as the histogram, the selection must be displayed differently - in this case with the bars of the histogram partially filled with colour based on counts of the events in the selection set (as shown in figure 4.6).

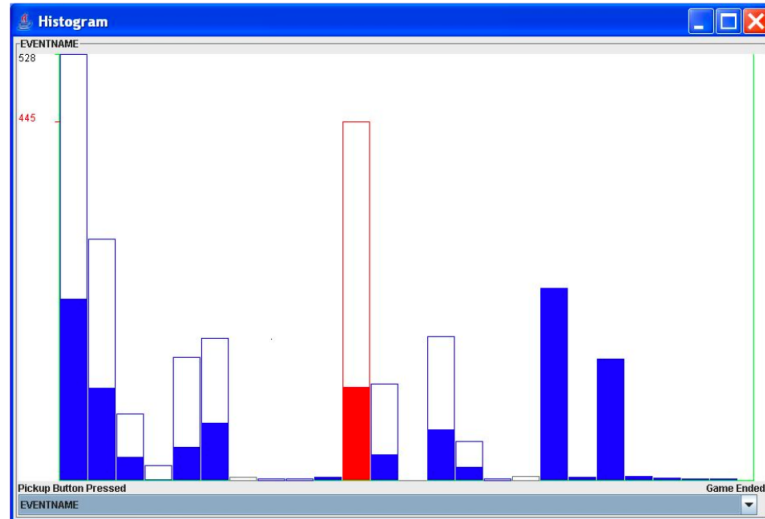


Figure 4.6: A partially selected histogram.

Each visualisation component is free to handle the data in whatever way its programmer sees fit. Replayer's architecture does not require the visualisation components to depend on any one model, simply that they send and receive data in the CSV format specified in the *iVisualisationComponent* interface. This allows for a diverse range of components

and techniques, with each using whatever storage model and visualisation technique is most appropriate.

### 4.3.8 Bridge Components

It is the case with some components that third party software already exists that does the required job effectively. In the current version of Replayer, these include Google earth for map based displays, and either QuickTime or Windows Media Player for displaying media content. It is however a requirement that Replayer's server be given some way to communicate with these programs. As such Replayer includes three 'bridge components', two media bridges, one each for QuickTime and Windows Media Player and a third for Google earth. As the bridge components are, like the visualisation components based on the *iReplayerComponent* interface they operate in much the same way.

#### Media Bridges

Replayer's two media bridges coexist on a one-per-platform basis. Those tools consist QuickTime on OSX and Windows Media Player on Microsoft Windows. The user interface and much of the functionality is identical between these two components, with only the layer that communicates with the program in question being replaced. Those layers will be examined presently, but first let us focus on the *Time Slider* component, which is replicated in both versions.

The time slider is a simple looking component. It comprises a Slider with a thumb, a user interface component that most computer users will be familiar with, Three buttons labeled 'Play', 'Stop' and 'Fix in DB' respectively. A start time label at the top left, and end time label at the top right, a current time text field in the top centre, and a master volume control on the bottom right. On startup, the time slider queries the server for the contents of the 'Media' database table. It also retrieves the state and event master tables. Next it finds the first and last timestamp from all of those tables. This gives the entire duration of the dataset, and defines the start and end points for the slider. It is possible to send the TimeSlider a reduced set of data by using the meta tool to send a smaller set of tables by making a constrained SQL query. Once it has the start and end points of the data set, the timeslider paints the slider's bar red in all the locations where there is media data available, based on the values in the media field. Additionally it stores a list of all the RIDs in the master tables, along with their timestamps in a list sorted by timestamp. As the thumb is



moved on the slider, when it enters one of the red areas, the corresponding media is opened in the appropriate program see the next section for details on how this works. Moving the slider will scrub these movies along, jumping to the correct time in the movie as the thumb progresses or reverses along the path bar. At times where a given movie is not supposed to be viewable, that is, when the thumb is at a time that that movie was not being recorded (according to the contents of the media table), the movies are hidden from view.

The time slider, like every other visualisation component, supports the selection process. Because it maintains a list of all the RIDs along with their timestamps, it is possible to select a slice of time in the interface, by dragging with the mouse, yielding a green colouration. The start and end times of this selection are then used by the timeslider to locate all the RID values within that slice of time. These values are then sent to the server as a selection and subsequently sent to all the other components. Received selections reverse the process, drawing single pixel green lines along the interface in all the places where events have been selected. The thumb, and subsequently the movies are then jumped automatically to the first of these lines, ready for the user to review the videos for that time.

Pressing the play button causes all the multimedia files to play simultaneously (single video playing is available through the normal media controls of the program in question). While playing those videos, the time slider also sends out selection information, once per second based on the location of the thumb. In this manner it is possible to ‘animate’ the selections in other visualisation components, yielding for example on a map the apparent motion of the users as a trial progresses, or the occurrences of events in an event series. Pressing the stop button, as one might imagine, causes this process to desist. The master volume slider simply sets the volumes of all the movies to the value defined.

Having considered the operation of the TimeSlider part of the media bridges we must now consider the area in which the two differ. First let us examine the QuickTime Controller. Apple’s OSX includes a scripting language called *Applescript*, which can be used to remotely control all of Apple’s own applications. As one of these applications, QuickTime is controllable in this manner. Contained within the Apple distribution of java, is a library allowing java to execute applescript programs, and thus it is relatively straightforward to instruct QuickTime to do exactly what one wishes. As such, when a user, for example, presses the play button, an applescript is executed which tells QuickTime to *Play All Movies*. The same technique allows data about the movie to be retrieved, so for the synchronization a script requests data from QuickTime about the current position of a movie. The same technique

is used to hide and show movies, and to jump to specific frames.

Windows media player does not share apple's scriptable approach. Fortunately there is an available solution. A small C# program, that runs in the windows *system tray* has been created. When a movie needs to be opened, it creates a window, and adds a windows media player COM component to that window, giving the user an interface, which if lacking the full features of windows media player, is certainly a familiar interface. This program communicates with the TimeSlider component over a TCP connection and achieves effectively the same result as the QuickTime alternative. Movies are shown, hidden, played, stopped and jumped as required by the user's interaction with the time slider.

### Google Earth Bridge

Google earth is a powerful tool that allows locations to be plotted on a globe. The 3D model of the earth is textured with satellite photographs of nearly everywhere on the planet, and can be zoomed in to a quality dependant on the highest resolution of photographs so far taken for that area. In some cases, particularly the larger western cities, those photographs are incredibly detailed. This then seemed like an idea companion to Replayer.

Locations can be plotted in one of two ways, either directly - requiring the user to click the location and tag it with some data or, alternatively, locations can be plotted using a Keyhole Markup Language (KML) file. It is the latter of these which is of interest to the design of Replayer. Google earth can be configured to get this file in one of two ways, either by opening it from the file menu, or indeed the operating system, or by retrieving it from an HTTP server on a predefined port and schedule. The highest resolution of these retrievals is 1Hz. This means that the fastest that Replayer can update Google earth is once per second, as there is unfortunately, as yet, no way to directly push data at the Google earth client.

The Google earth bridge component starts up exactly like a normal Replayer component, and has a simple user interface that allows the user to make selections about how the resulting KML file should be structured. A set of drop down menus allow the user to define, from the columns made available by the SQL statement generated in the meta tool, a main variable - which is the text that will appear in Google earth as the title of that object, an x position, a y position (the bridge component is able to automatically and transparently convert between British OS and latitude/longitude values, and finally a variable by which the values will be coloured - this allows an extra dimension of data to be represented at a glance. An example of this setup from the treasure game described in the next chapter, might be the Player's

name as the main variable, the x and y coordinates, and then the glyphs coloured based on the player's team. Note that it is acceptable for the colour variable and the main variable to be the same. Additional variables can be added by selecting them from a list box on the left side of the UI. These are added to the KML and represented in a pop up box when the user selects one of the objects in Google earth.

Once the setup process is complete, it generates that KML file and makes it available to Google earth by running its own private HTTP server on a user defined port. Each time a selection update is received from the server, the KML file is re-generated, either excluding the non-selected objects completely, or colouring them grey, depending on the user's preference.

When the user starts the Google earth application, they must select the options from within the Google earth user interface to get the KML file from Replayer's server. This requires them to input the IP address of the computer running the bridge component - typically the same machine, so the loopback address 127.0.0.1 is the most common and the port, typically 8081. Once this is done, they must select a refresh rate.

One of the biggest problems with the choice of Google earth as a visualisation tool, is that there is no obvious way to retrieve the selections a user makes from the program and send them back to Replayer. The only option available, comes from a quirk of Google earth's HTTP GET request. When it asks for the KML file, it sends the enclosing coordinates of the image currently shown on the screen. This means that it is possible to use this box as a crude form of selection. When the GET request is processed by the HTTP server, it passes that data into the bridge component, which can then use a simple algorithm based on Java's `2DRectangle` object to establish which RIDs are within that enclosing rectangle. This is considered to be a selection and passed to the server, which then passes it on to the other components.

The Google earth bridge is at best an imperfect solution to the problem of plotting positional data for Replayer. The selection process is clunky and the updates are potentially too slow at 1 per second. One option for updating this component to solve these issues might be to use the Google Maps API instead and display data using HTML and either a default browser or a custom viewer component.

### 4.3.9 Extensibility

Each Replayer client is a separate program in its own right. It need only adhere to the communications API to be able to communicate with the server. As such, there are no restrictions on the way it is implemented. To make it accessible to the user however, it must be packaged up into a format that the meta tool is able to examine, and thus display in the ‘start new component’ screen - though if this is not done, one can still use such components by opening them from outside the Replayer environment. This packaging requires them to be stored in a java archive (jar) along with the path for the main class, and a screenshot which the meta tool uses in its menu. This means that a simple java wrapper which executes a *Runtime.exec()* command will be required for non-java based clients.

Additional clients can be added at runtime as the ‘open new component’ menu in the meta tool scans the folder where clients are stored every time it is opened. The intention here was to create an architecture which would support third party components, encouraging a system whereby if a particular component is required, it can be added by someone with a little programming experience, then shared with a larger community. In theory, a community of Replayer users could generate a pool of available open source components and one would be able to search this for suitable components before having to create a whole new one.

### 4.3.10 Digital Replay System’s Client-server system

The DRS desktop application can be used either in a standalone mode or a client mode, associated with a particular DRS workgroup server. DRS’s support for collaboration and distribution are only available when used in client-server mode.

When running in standalone mode all projects are stored exclusively on the local computer. However, when running in client-server mode each project can be either standalone (local only to that DRS client) or server-based (and potentially accessible to other clients, depending on permissions). A standalone project can be converted to a server-based project at any time (but not vice versa). In a server-based project it is up to the user to decide which files should be uploaded to the server, and which maintained locally only (and/or distributed by other means than the DRS server).

#### 4.3.11 Server Overview

The DRS server, like the client, is written in Java. It runs as a J2EE (Java 2 Enterprise Edition) web application, in a servlet hosting container such as Apache Tomcat. It has two interfaces:

- An XML Web Services interface, which clients use to communicate with the server (based on Apache Axis), and
- An HTML browser interface, for DRS users and administrators (based on the Spring Framework and Java Server Pages).

Like a DRS client, the server uses persistent JENA RDF models (backed by a Hypersonic SQL database) to hold an index model and the metadata for each project. The server also has a local file storage area for project-related files, and can upload files from and download them to individual clients for local use. Project-specific databases do not exist on the server; it is assumed that they can be regenerated by clients using the log files and data processor definitions which are stored on the server.

#### 4.3.12 Security and access control

The DRS server is presumed to be secure - it is up to the local system administrator(s) to take the usual precautions to protect the machine and access to it. The full RDF of all (server-based) projects and any project files that have been uploaded are all accessible to the server. However DRS clients need not be assumed to be secure, and different DRS users may have access to only a subset of server projects (and, in principle, only a subset of the RDF and files within any one project). The main point at which this is policed is in the interaction between the client and the server. The DRS server website is the starting point for security and access control. When first deployed there is a single pre-configured administrator for the server. This bootstrap user can then add other users (as server administrators or regular users). The web site is also used by administrators and project owners to control which other users have access (read and/or write) to which projects.

#### Client Identification and Authentication

A new DRS client must be added to the server by an administrator - an administrator user name and password are required by the client during the initial client registration process

when the client is first run on a new machine. The registration process establishes a shared secret (a random string) between the client and the server which is used to authenticate that client in subsequent interactions (excepting un-trusted clients). DRS supports a number of different strategies for authenticating users, according to the nature of the machine that the client is running on; this is configured by the administrator when the client is first registered with the server. The options are:

- Client machine is un-trusted - this requires an explicit user log-on every time the client application is used, and should typically avoid retaining local project metadata or files after the session is over. Note: this is probably unsupported at the present time.
- Client machine is trusted and single-user - this does not require a log-on or authentication from the user each time the client application is used, as it assumes that the normal (physical and OS) security of the client machine are sufficient to reasonably identify and authenticate the user.
- Client machine is trusted and shared (typically part of a multi-user domain) - this does not require a log-on to the application itself, but the user identity is mapped from the current operating system identity, i.e. it is again assumed that a successful access and authentication with the OS of the client machine is sufficient to authenticate the user (and that the OS user accurately identifies the user).

There is no fundamental reason for a DRS user to actually be a single person - if a group of people normally shares use of a single OS user account then they will normally also appear as a single user to DRS.

### **Server Access Control**

Almost all client requests to the server are subject to authentication - of the client and user - before handling (the only exceptions are checking the server version and initial authentication). Consequently, when each request is actually processed the identity of the requesting user is known. DRS currently has a relatively simple access control model: each user can have no access, read access or read-write access to any one server project. The server index model is used to persist access permissions, and this can be changed via the server web pages (which require the user to log into the web pages independent of their use of a DRS client). The server will only provide new project data or files to a user if they have read permission at the time of asking. Similarly, the server will only access project updates

or new file uploads if the user has write permission at that time. If a user's permissions are reduced then they will normally still have access to the copies of project data and files which had previously downloaded from the server, and any data which they had previously contributed will still be present in the project. By default, a new project is readable and writeable only by the user who created it. They can then grant read or write access to other users registered with the workgroup server. In addition to read and write access, the server also keeps track of granted access, i.e. the right to change other users' access to a project. A DRS server administrator has grant access on all server projects (e.g. in case a user leaves the organisation). Otherwise, as with read and write access, the user creating a project is the only one with initial grant access, although they can grant this same facility to other users.

#### 4.3.13 Client Access control

The DRS client maintains a separate RDF model for each project. More precisely the client maintains a separate RDF model for each user's own view of each project. So if the same client machine (and DRS client application) is used by two different users, then each one will be using a disjoint set of RDF models, even if they are working on the same server-based projects. This provides a basic level of access control on the client, since the client will only give the user access to their own copies of the project data. So if a particular user is denied access to a server project by the server they will never be able get their own copy on the client. However, at least in the current implementation, these models are actually part of the same database, to which the client application has full access. So a sophisticated user (programmer) could gain access to the project views of other users on the same machine. We assume that local policies and working practices (including trust) will be sufficient to address this issue trusted multi-user machines (but it follows that highly sensitive data should never be accessed at all from a machine that is not itself sufficiently secure). Depending on the configuration of the client machine and its operating system it may not be possible to prevent other users accessing local video and log files stored on the machine (e.g. through the normal operating system tools and applications). However, users are able to make use of any operating system facilities which are available to guard the files that they are working with from other users of the same machine (e.g. OS file and directory permissions), since they will normally run the DRS client from their own user account (with their own OS identity and rights for file access). At present, files downloaded from the server are cached

in a common client directory, and so would be accessible to other users of the same machine. A more secure option would be to rely on personal removable storage (e.g. external hard disk, flash memory or DVD) which is only plugged in to the client machine when those files are actually required and can otherwise be independently physically secured.

#### 4.3.14 Project Synchronization

The data for each project is a single RDF graph (plus associated files and databases on disk). For a server-based project, the server is considered to hold the complete and up-to-date version of this project data. Each project has a server version number (an integer), which is incremented each time the project is updated on the server. When a new project is created it is initially a standalone project, known only to the creating client and user. The user can then work on that project locally if they wish. At this point or at some later time the standalone project can be converted to a server project. At this point:

- the client uploads its complete RDF model to the server,
- the server:
  - creates its own new persistent RDF model for the project and loads the client's data into it,
  - checks and updates the project information (e.g. creation date, version),
  - copies essential information about the project to its own index model,
  - grants initial project access rights to the creating user,
  - creates a new RDF graph to the client reflecting the state of the server project as that user is allowed to see it (often but not always this would be the full model),
  - stores a copy of this on the server, so that it has a persistent record of what the client was given of the project metadata,
  - and returns the data to the requesting client.
- The client then
  - replaces its old (uploaded) project data with the new data,
  - stores a copy of this locally, so that it also has a record of what information the server gave it, and
  - updates its own index information (e.g. with project server version number).



Note that the server also caches a copy of the RDF data that it returns to the client, so that it has a persistent record of what the client was given of the project metadata. This is illustrated in figure 4.7

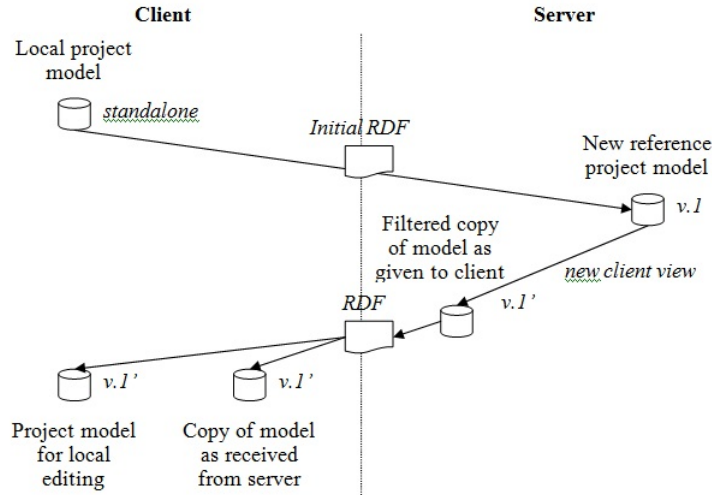


Figure 4.7: Creating a new server project from a standalone project.

#### 4.3.15 Synchronising client and server

Each user on each client machine has a separate copy of a server project's RDF metadata (filtered according to any read access constraints for that user). Each client also has a copy of the project metadata as originally received from the server and without any local changes. Whether a user has write access to a project or not they can make changes to their own copy of the project on their own client machine like any other (server or standalone) project - no communication with the server is required for normal DRS operations. This is an important design goal of DRS client-server approach, and allows the analyst to work effectively when away from their home network (or, indeed, any network, for example while travelling). When a user has done some local work on a project and wishes to update the server they choose explicitly to synchronize that project with the server. If successful, this will make those local changes to the metadata available to other project users and/or from other client machines. Synchronization has two phases.

### Client Update

First, the client must get itself up to date with respect to any changes (by other users or from other client machines) made since this client/user version of the project was downloaded from the server. It does this by requesting an update from the server. The server:

- Checks that the user still has read access to this project;
- Makes a new snap-shot of the server's version of the project metadata, filtered according to any access restrictions that this user has,
- Retrieves its copy of the version of the project which it previously gave to this user and client,
- Calculates the difference, i.e. which RDF statements have been added and which removed since the last update,
- And sends these differences to the client.

The client can then apply these differences to its own local model. If the changes that have occurred on the server do not overlap with those just made on the client then these changes will all be simple to make - just adding and removing statements. The client also uses the changes to update the copy that it has of the project data as originally received from the server. This is illustrated in figure 4.8. However, if the user has changed some of the same

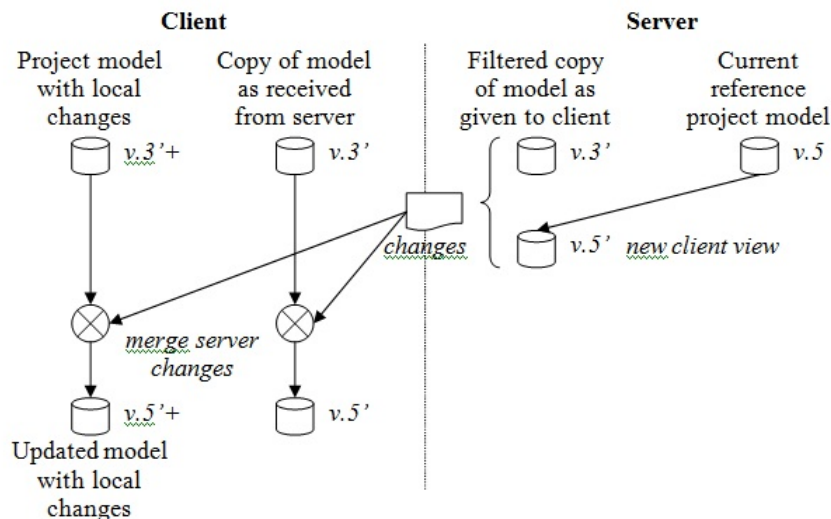


Figure 4.8: Client update process to merge server changes.

things then the changes from the server may no longer make sense. For example, two users

might change the title of the same analysis concurrently. The first might then upload their changes to the server. Then when the second user tries to update they find a request to (a) remove a statement that is no longer present (the old title) and (b) add a new statement which conflicts with one already present (a second value for a functional property - the new title value). There is no general automated solution to reconciling these kinds of concurrent changes. The current implementation has some simple heuristics, e.g. checking for functional properties (or property restrictions with cardinality 1) and retaining the locally changed value in preference to remotely changed value. However this is not a complete solution, and more needs to be done in this area (for example, automatically merging changes in textual content, and/or referring conflicts to the user for manual reconciliation). Most problems of this kind can be avoided by appropriately designed working practices. For example, if each users creates and edits their own analyses then they are unlikely to make changes which conflict with others. If a client merges its updates successfully then its local model will now be up to date with respect to the server, but still contain the additional changes that had been made locally.

### Server Merge

Once the client merge is complete, the client can then perform the second phase of Synchronization, which is to pass those local changes back to the server, to be merged by the server into its reference version of the project. This is essentially the same as the client update process but with client and server roles partially reversed:

- The client
  - Retrieves its local copy of the model as received originally from the server (incorporating any updates received from the server, above),
  - Calculates the difference between its current (edited) model and this, i.e. which RDF statements have been added and which removed locally,
  - Sends this to the server.
- The server
  - Checks that the user has write access to this project, rejecting the request if they do not, otherwise
  - Applies the client's changes to the reference version of the project,

- Increases the server project version number (changes have been made),
- Makes a new snap-shot of the server's version of the project metadata, filtered according to any access restrictions that this user has,
- Retrieves its copy of the version of the project which it previously gave to this user and client,
- Calculates the difference, i.e. which RDF statements have now been added and which removed (at the client's request) since the last update,
- And sends these differences to the client.

The client can then apply these (approved and visible) differences to its local copy of the model as previously received from the server. This is now copied into its current view of the project. This is illustrated in Figure 4.9. Note that there should be no conflicts during

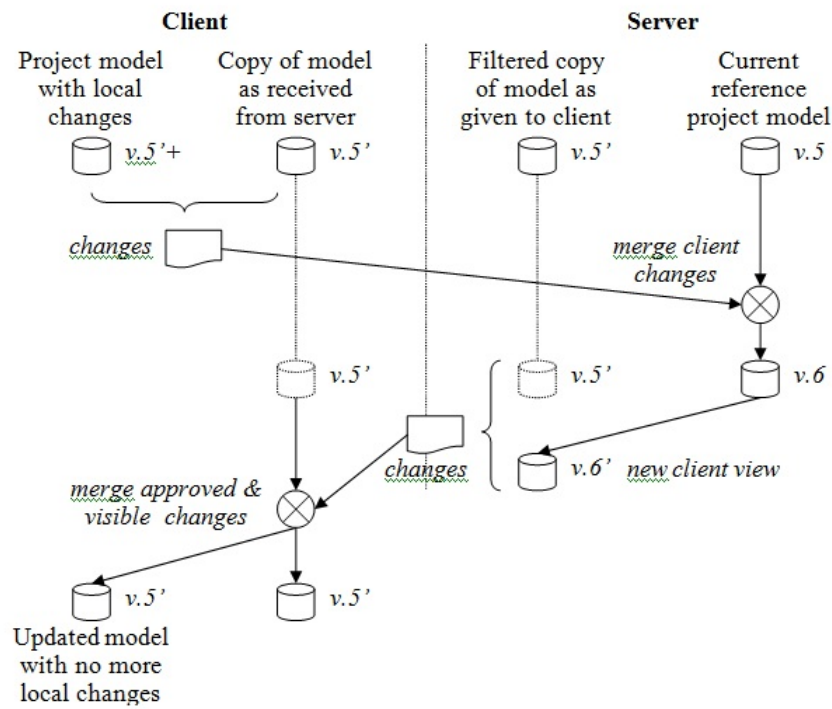


Figure 4.9: Merging client changes with the server.

this process, either on the server or on the client, since the updates from the client are now based on an up to date version of the server's information, and the updates from the server are based on a known model in the client.

#### 4.3.16 Comparing the two approaches

While Replayer and DRS both demonstrably use client-server approaches, the goals, and thus design and implementation of these approaches differs drastically. DRS, the more heavily engineered of the two projects takes a monolithic approach to clients. That is, the client is a single application per computer with a number of viewers and editors for the data. This limits collaboration to some extent, making it a solo application, but the server extensions allow small teams to work on the same data. Ultimately, this approach is the more scalable of the two, as it is possible for a single client to have several servers with different data stored on them. It is also significantly easier to set up, as it has a single-user version which entirely bypasses the need for a server and is sufficient for most applications. A spinoff project of DRS called Mixed Media Grid (MiMEG) created a cross-site collaborative version of the system however [82].

By keeping each viewer as a separate process, in effect a whole separate program, Replayer allows users to more effectively tailor the tool to their needs. Setup is more complex as a server is required to run any session, even on a single machine, however once set up Replayer benefits in a number of ways from its approach. First, Replayer does not require the kind of window organisation tools that DRS does, as these are embedded in most modern operating systems. Second, a Replayer analysis can be spread across several machines on a network which allows two key benefits: collaborative analysis and additional processing power/screen-real-estate. While DRS is limited to the maximum processing power of a single computer to run all its viewers, and thus has a hard limit on the number of those viewers that can run simultaneously (especially in the case of video playback), Replayer is able to share its processes between two or more computers, circumventing those limits (though Replayer Synchronization between two computers is compromised by network lag). In discussions with users it has been shown that it is really the last of these benefits that appeals from Replayer. We with our pilot users, we saw few examples of collaborative analysis, though MiMEG has demonstrated that there is a recognisable demand for it. The final point to consider is that of extensibility. Both programs offer facility for this, but of the two, Replayer offers a significantly more versatile model. DRS requires extensions to be written in Java, implementing the (fairly complex) viewer interface and for them to be registered with the main program. Replayer on the other hand has no set requirements other than that the module is able to read and write to the correct TCP ports and uses the Replayer communication protocol to send and receive selection data. This increases the number of

people able to create extensions to Replayer, simply by removing the requirement that code be written in Java. Of the two tools DRS is the most immediately familiar to its core user-group (social scientists) since the suite of tools it delivers are more recognisable to users of existing CAQDAS software, however several of those users find the program excessively complex because of the sheer number of facilities it provides. While Replayer can seem like a simpler alternative, it lacks the more traditional CAQDAS tools, such as transcription and a coding system. It is worth noting that in a proposed rebuild of DRS the architecture will be brought closer in line to the Replayer model. One area in which the DRS model stands far superior to Replayer is that of multiple project support. Replayer can handle exactly one database at a time. While that database can easily be saved, exported and passed around, it has nothing like the kind of corpus support enjoyed by DRS. Indeed, as will be seen in the proceeding section, DRS is even able to handle multiple log file databases to provide comparative views of different studies within a single analysis - something Replayer is simply unable to do. Likewise DRS's server system allows users to have data stored securely on several servers. Replayer is simply not capable of that kind of sharing. Despite the fact that of the two, Replayer is billed as the collaborative analysis tool, usage has shown that it is the type of collaborative work supported by DRS, i.e. users working independently but sharing a corpus of data, that is more practically useful than the kind of simultaneous data analysis that Replayer supports.

## 4.4 Synchronization

### 4.4.1 Introduction

It is important to note here that when discussing synchronization we mean more than just the technical task of keeping data temporally synchronized, though that does form part of the discussion. There is also the idea of methodological synchronization. How heterogeneous media can be 'used' together.

### 4.4.2 Synchronization in Replayer

Synchronization is a fundamental feature of a system such as Replayer. Without accurate synchronization, one cannot easily relate one data set to another. There are two different types of synchronization used in Replayer which we must consider. One is the synchronization of multimedia clips with each other and with the system logs. The other is the internal

synchronization required to implement the brushing and linking described in the section on viewers.

First let us consider the case of multimedia synchronization. It is necessary for each device creating a log to be synchronized in advance of a trial. This gives internal synchronization within the logged data, however, synchronising video and audio tracks with the system logs is more complex. We use a lo-fi technique based around a tool called QCCI (Quickie). This is an application designed to be run on a PDA given to each evaluator of a system in use, i.e. each observer who goes out into the field. It displays a clapperboard image on the PDA screen, and the time on the PDA is synchronized to the other devices through wireless network connections.



Figure 4.10: The QCCI synchronised digital clapperboard

From a user's point of view synchronization points are set in the timeline slider. A button is used to synchronize new movies with the logged data. To achieve this synchronization, it is necessary to open the desired movie and find a time where there is some obvious connection between a the system time and a given frame in the movie. The easiest way to achieve this is if the recorder of the movie has used a system such as QCCI, wherein the observer's video camera is pointed at the screen of the QCCI device, which displays an accurate system clock, that has been synchronized with the devices being logged. This gives a user the system time displayed on a frame of the movie. The user can then enter that time in the current time field, resulting in the thumb jumping to that time, thus coordinating the frame of the movie with the correct system time. The timeline slider sends a message to the server to write a

new time into the media table, along with the file path and details of the currently selected (topmost) movie. That movie is then permanently synchronized with the data and can be used as described above. If QCCI or a similar technique has not been observed, the user must find some event, usually using the event series that is obviously observable in the movie and base the synchronization on that.

One slight problem with this technique is that the best possible synchronization is based on the framerate of the movie, so a movie recorded at fifteen frames per second can at best be synchronized accurately to one fifteenth of a second. General use showed that synchronization tended to be out by anything up to a second. Additionally, some forms of compression - notably MP3 played back with QuickTime, slightly stretch or compress the file, meaning that the synchronization will drift the further one is from the synchronization point.

This is a problem about which little can be done, though if the compression amount were known and constant it would be possible to repair it. Imperfect synchronization is spotted most readily in audio. Playing back two media files that are imperfectly synchronized results in a slight echo. It has been observed that the best way to handle this is simply to mute all but one of the movies. Interviews with users have also suggested that it is difficult to seriously attend to more than one movie at any given time. This does not in itself mean that it is not desirable to have multiple simultaneous video streams. Indeed, many of the points laid down in the challenges in the introduction call for multiple views on one's data. Just because a user attends to only one at a time does not mean that the others are not part of the analysis. A user may examine an area of video, then immediately want to review what happened at that time in the system logs, or in a video of a non-located user. Realistically such examinations happen one at a time however, so for most applications imperfect synchronization is not considered a major issue. In one interview, the analyst said that she liked to watch the whole lot together first, just to get a general 'feel' for the data, then began to focus on the detail of each part of the data individually, and examine the relationships between those different data streams with a view to describing a particular accountable event.

Audio synchronization is handled similarly, using a distinctive noise that is recorded on the audio track and simultaneously logged. This technique does not actually require the use of QCCI, indeed any suitably accurate display of the time is acceptable, as long as that time is synchronized with the system clocks of the recording devices, or at least one is aware of



the offsets involved.

Beyond synchronising logs, videos, audio, the second requirement is to keep the data selections synchronized. With many different visualisations of heterogeneous data types, the challenge of keeping everything synchronized is considerable. There are two specific areas of the system where data synchronization must be maintained. The meta tools must remain synchronized 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.

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. synchronization 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 multicasts a list of currently active components. Included in this list is 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 displays appropriately.

The server is responsible for facilitating brushing and linking process between visualization clients. Thus, when a user makes a selection in one of those clients it must notify the server, which must in turn notify the other clients. Here we see the true purpose of the RID value in each row of the database. A row of the database can be considered to be an object, in that it refers to either a discrete event, or a slice of time where some state value is the case.

A selection in Replayer consists of a collection of RIDs with each RID referring to the object represented by that row of the database. Some clients, such as the time series and the media bridges may appear to be selecting a slice of time, but in fact they are selecting a collection of objects and it is this that is sent to the server. The server maintains a list of currently selected objects which is then multicast to the current collection of clients. A selection message uses the keyword `$SELECTION` then a comma separated list of RIDs. When the server revives one of these lists, it is straightforward to simply reflect this list back out to the clients. However, this is not sufficient to achieve the intended task.

Let us use an example to illustrate the problem. Using our earlier example from the database architecture section, we have event data pertaining to button presses and state

data pertaining to the locations of the users. Now consider two views on this data. We have the all the location traces plotted on a map using the Google earth bridge component, and the button presses plotted on an event series. We want to plot just the positions of those button presses on the map. If we select all the button presses on the event series, we see the disappearance of all the plots on the map. This is an effect of the RID method of object selection - we have selected all the button press events, by their RIDs and broadcast that list out as the currently selected list of objects. However, because the map is displaying the state objects, which have a completely different set of RIDs we see nothing selected on the map. The same thing would happen if the situation were reversed. If we wanted to see all the button press events that occur within a constrained area of the map and made that selection using the map tool, none of the button press events would be highlighted.

The server, on receiving a list of RIDs examines each one individually. If it is an event, it uses the timestamp to find the closest preceding recorded value in the state data and includes this RID in its broadcast response. Note that it must be the preceding value, even if the proceeding value is closer as otherwise the data are certainly untrue, while this is not perfect, it does return the best possible known state data for the time that event occurred. If the selected RID is a state value, then the server will look for pairs of values - that is two values in the selection set which immediately proceed each other in the state table. If it finds a pair, it uses the timestamps of that pair to select all the event objects that occur in that slice of time. Filtering of these events must then be actuated on the client side. In figure 4.11 we see state entries and event entries shown over time (y-axis). The red box shows how when selecting a slice of time including two or more state entries, *all* events within that timeframe regardless of type will be selected. The figure also shows the last known state for each event. When an event is selected the last known time is also selected (not the closest time).

In this manner it is possible to make and maintain selections which cross the boundaries of the underlying state/event model.

Effectively then we have a system of data synchronization that can be applied through any dimension of the data, rather than exclusively temporal.

### 4.4.3 Synchronization in DRS

From a technical point of view, the current incarnation of DRS handles all its synchronization by time. DRS supports a simple synchronization model: each file in the (analysis) file-set has

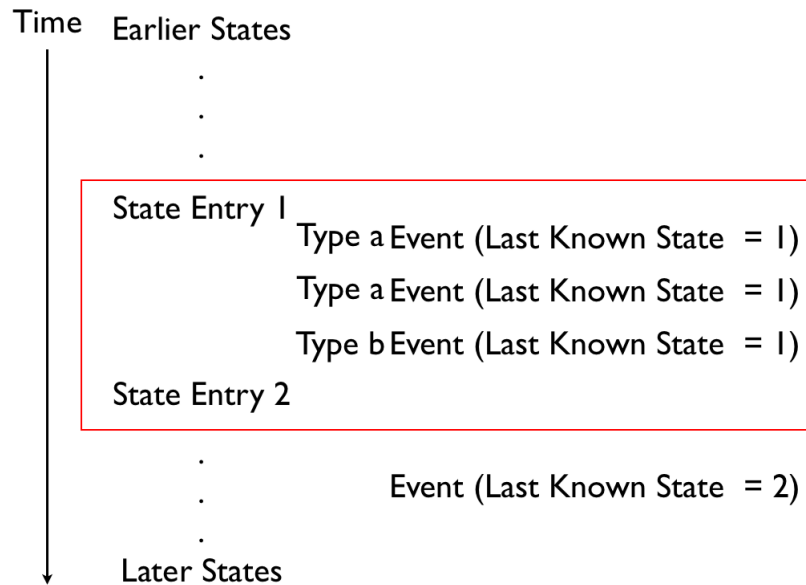


Figure 4.11: Interrelation of state and event selections

its own start time relative to the file-set as a whole. So playing from time zero (for example) in the file-set would actually mean playing from the individual start time of each file within the set. This start-time could be set by finding a distinctive visible or audible event (a ‘clapperboard’ event) in one file and temporarily freezing its playback, playing the other file(s) to get to the same event, and unfreezing the first file. DRS supports this functionality through the analysis *Synchronization Manager* - but with scope to significantly extend and refine it, by the definition of time classes within its ontology. File time offsets can also be adjusted by graphically dragging media tracks in the Track Viewer. Time is modelled in the digital record ontology by the ontology class `dr:Time`. This is specialised into the two subclasses:

- `dr:AbsoluteTime`, which specifies a complete date and time (e.g. 6th June 2006, 12:03:02pm GMT) as determined by some kind of timing device such as a computer or camera with a real-time clock, or a wrist-watch or wall clock, and
- `dr:TimelineTime`, which specifies a time by an offset along some abstract or concrete ‘Time line’ (modelled by ontology class `dr:Timeline`).

The ontology models several different ways which might be used to specify time-line times:

- The unified time within a single analysis or replay (ontology class `dr:EpisodeTimeline`),
- Time within a single media file, i.e. time from the start of a video or audio file (ontology

class dr:MediaTimeline),

- Time as measured by some timing device starting from some (absolute) start time (ontology class dr:TimingDeviceTimeline), e.g. ‘UNIX’ times, which are typically seconds since midnight on the 1st January 1970, or
- Time based on some offset from some other time-line (ontology class dr:RelativeTimeline).

Every analysis has exactly one timeline - its dr:EpisodeTimeline - and all replay is based on this. Every time-based media file (such as a video) has its own dr:MediaTimeline. The relationship between replay time in the analysis and time in each file viewer is determined by the mapping from the current time in the episode timeline to each viewer’s file’s media timeline. If the analysis and the media both have known absolute start times then these can be used to synchronize playback (see Figure 4.12).

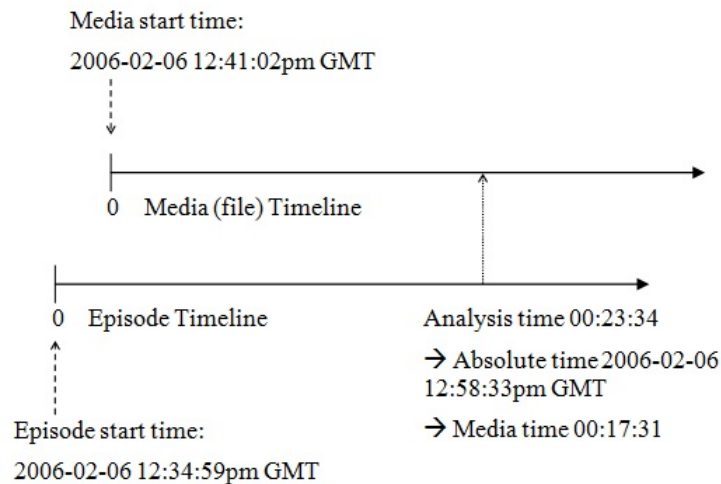


Figure 4.12: File/analysis Synchronization using absolute times

Otherwise (or in addition) one or more dr:TimeRelations can be used to specify points of correspondence between times on the different time-lines, such as the ‘clapperboard’ events already mentioned (see Figure 4.13).

It is theoretically possible with the current model, though not implemented, to specify multiple time relations at different point in the respective time-lines, e.g. to compensate for differences in relative clock speed, or the pausing of one of the time-lines (e.g. a video which has been paused and resuming part way through recording). While there are good reasons for modelling these details of time and synchronization, it does raise significant additional challenges for user understanding and for creating a reasonably simple and usable application

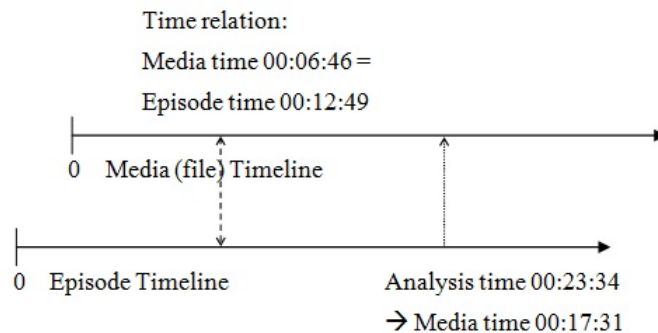


Figure 4.13: File/analysis Synchronization using absolute times

interface.

### Methodological Synchronization within DRS

Of course the technical challenge of synchronising media along a timeline is only part of the idea of synchronization within a piece of CAQDAS software. Once the data is temporally synchronized we must consider what can actually be done with it. One area to examine is corpus management. This may be seen as a form of synchronization. DRS provides a method by which all different types of media may stored as a single multimedia and multimodal corpus. The project and analysis system allows these data to be not just stored but organised into sensible subsets can be synchronized via the client server system with other users.

Second we should look at the familiar CAQDAS tools provided by DRS. First we have transcription, coding and annotation of time based media, all of which may be considered a form of methodological synchronization. In each case we are essentially creating meta data which can be effectively searched via the concordance tool, thus though we rely on time to take us to the correct point in a related media item, searching our meta data allows analysts to look for connections in the data, and indeed exporting that meta data allows more sophisticated statistical methods to be applied using tools such as Microsoft Excel and the Statistics Package for Social Scientists (SPSS).

The other area of interest here concerns the DRS document tool. With this tool we are able to create *active* documents allowing data that has no inherent temporal information to enjoy an asynchronous link with the temporal media. As an example we might have the transcript of an interview conducted as part of a study. The interviewee may reference events

recorded elsewhere on the timeline. The system of coding within the DRS document allows a user to link that reference back to the actual referenced event, so it can be examined with a single click. Indeed the system is such that several events happening at different times may be referenced simultaneously within the text and each reference can easily be mapped.

What is essentially being done here is the addition of extra contextual data to a document. In practical situations, an analysis is the process by which a descriptive document is built. Taking ethnography as an example of the kind of qualitative analysis DRS and Replayer are designed to support, a professional ethnography is described in terms of a formal analytic schema for intense analysis, and packaged in a report [88]. DRS's document viewer provides a means by which that report can be augmented during its design phrase with what amounts to quick reference bookmarks to the data drawn on by the ethnographer.

#### 4.4.4 Comparing the two approaches

From a technical point of view, both Replayer and DRS' temporal synchronization at playback time is simplistic at best. All that is made is a 'best try' to start viewers playing at the same time. MiMEG demonstrated that when using networked simultaneous playback this is simply insufficient, and demonstrated technology by which an improved playback synchronization may be achieved. No provision is made in either system to cope with the idea of 'drift' wherein two items drift further and further out of sync as playback time increases. This seems to be particularly problematical in DRS when playing back audio files encoded as MP3s.

Likewise the media synchronization is essentially very simplistic. While log data synchronization is limited only by the maximum accuracy of the least accurate clock used (assuming those clocks have been synchronized in the first place) or by any delay between starting logs recording separately, video synchronization is limited first by the framerate of the video(s) in question, second, by the accuracy of a recorded clock (if a system such as QCCI is used), and further by human accuracy, if using a more traditional clapperboard (audio based) system. One solution to this problem, currently being explored in forthcoming updates to DRS is the augmentation of cameras to include accurate timing data about when recordings were started. The only major problem with this approach is it requires specific hardware, while both systems are otherwise usable with exclusively commodity hardware. Replayer actually takes this idea in a slightly different direction as described in this paper [158]. In this proof of concept work, the cameras were augmented with location and bearing recording

systems. The subsequent generated logs could be used to determine where a given camera was pointing at a given time, and thus determine ‘usable’ footage. In the example in the paper ‘usable’ footage was defined as footage where the camera was sufficiently steady to be viewable and more crucially, particularly for ‘static’ cameras, when one of the study participants was actually in shot (determined by a GPS log of the participants location and the effective viewport of the camera).

## 4.5 Conclusion

In this chapter we have reviewed the differing approaches taken by Replayer and DRS in three key areas of system design: distributed architecture, log file handling and synchronization. In each case there are some general similarities, but the direction taken by the two systems demonstrates the differencing approaches used in their development. Replayer, which was developed centred far more on the system log data, generally revolves around support for that, with additional media taking a less central role, while DRS is aimed more squarely as a familiar CAQDAS tool, providing the sort of functionality (transcription, coding etc) more normally associated with those tools (as can be seen in the related work section). As yet DRS has not focussed on ‘doing’ much with system log data in any generic way at least, though as we will see in the next chapter, when a bespoke log file viewer is created, DRS demonstrates how that data can add significant value when performing a qualitative analysis.

## Chapter 5

# Case Studies

### 5.1 Introduction

This chapter will present two separate case studies. One of the use of Replayer to study the mobile game *treasure* and the second on the use of Digital Replay System, also studying a mobile game: *Day of the figurines*. In the first case the study is performed *in house* by the Treasure development team, while the study of Day of the figurines is performed by a professional ethnographer from outside the project. In each case we will see how the use of coordinated views of system log data is used as a resource to describe in detail interaction details which could not otherwise have been captured, thus demonstrating the value that such resources can add to a qualitative analysis. It is important to note at this point that while these case studies provide good anecdotal evidence of the benefits of using the tools described in this thesis, this should not be considered a formal evaluation of those tools. Rather they will serve as examples to drive discussion.

### 5.2 Treasure

The first case study is based on data collected during trials *Treasure*, a mobile game developed at the University of Glasgow to explore issues of seamful design [47]. The main aim of a Treasure player is to collect *coins* placed in areas of poor or non-existent network coverage, and then bring these coins back into an area of good network coverage, and upload them to gain points. By moving in and out of areas of network coverage, players also survey the wireless network they are playing in, building up a dynamic map of network coverage that



they all share.

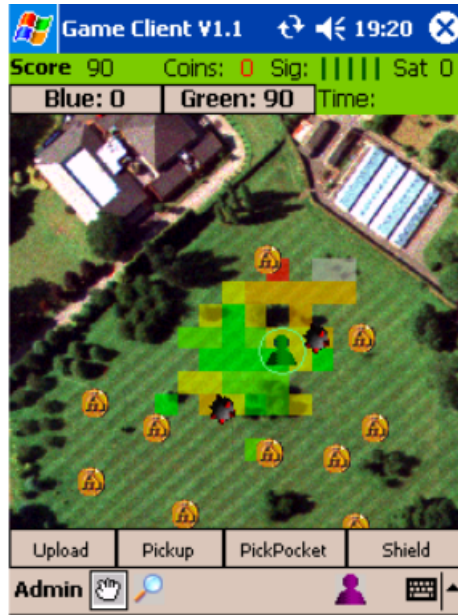


Figure 5.1: The PDA interface to Treasure.

At the beginning of the game, each player is given a Personal digital assistant (PDA) with GPS and 802.11 wireless capabilities. To pick up a coin, a player must walk or run to the physical location of the coin as indicated on the map, so that his or her GPS-tracked location is close to the coin's location, and then press the 'pickup' button. For the player to gain points for this coin he or she must then walk or run to an area of sufficiently high network signal strength and click 'upload' so as to send the coins he or she has collected to the game server. The chances of a successful coin upload increase the deeper a player is inside wireless network coverage. To be successful in the game, players must therefore learn which areas are covered by wireless network and which are not. In other words, they have to learn and use the *seams* of the game infrastructure. A key competitive game feature is *pickpocketing*. When players are close to each other they can use the 'pickpocket' button to steal coins that are being carried by other players. However, for a pickpocket to work, both players need to be within network range i.e. one can gain safety by staying out of the network. Players can also protect themselves from such attacks by deploying a 'shield', preventing other players from stealing coins that they have collected. Those players bringing coins into network coverage have to be aware of where their opponents are, keeping a distance or shielding themselves so as upload coins before they are stolen. In addition to coins, 'mines' are occasionally placed in random locations on the map. When a player walks over a mine,

their PDA vibrates and is disabled for twenty seconds. This also causes the player to drop all the coins he or she was carrying, and prevents the player from participating in or observing what is going on in the game via the game interface.

Figure 5.1 shows the interface to the game. At the top of the screen, information is displayed showing personal and team scores; coins currently being carried; signal Strength for both WiFi and GPS and the remaining time in the game. The map (in this case an aerial photograph of the play area) is displayed with notation for the location of each coin and mine; the location of the player (and any other players within range); the range for pick-pocketing (ring around the player); and an aggregate map of the wireless network strength (shown as alpha blended blocks). At the base of the map are the interface buttons allowing the player to upload coins, pick up coins, pick-pocket opponents and shield themselves from mines and pickpockets. Finally there are buttons allowing the player to switch between zooming and panning the map, and to centre the map on their location.

The game was played in teams of two versus two. If team-mates upload their coins to the same access point within two seconds of each other, they gain double the normal point allocation. A wireless coverage map was constructed dynamically by the server, from coverage data sampled in real time by the players, and was regularly broadcast as part of the game state for display as a semi-transparent map layer on users' PDAs (see figure 5.1). Green squares show areas of high sampled signal strength, and yellow squares show areas of weak coverage. These collaboratively constructed maps provide players with additional awareness of the network strength in the game environment, and also reveal where players have been and can be used to select suitable places to upload coins and areas where pickpocketing is likely to work.

In [14], Barkhuus et al. examine the evolution of tactics in *Treasure* as teams played the game several times. Recording of data in *Treasure* was threefold: log data was collected from each PDA, and separately from the server; videos were recorded of the game from two locations, one roaming camera in the field, and another from a vantage point in a window overlooking the game area. Finally, after each trial, participants were interviewed about their experience. The data was loaded into *Replayer*, and then the results of the interviews were collated with the recorded data.

### 5.2.1 Inaccurate Interface Displays

One Issue, inherent to the *seamful* nature of the game was the fact that a player's PDA display of the other participants' positions seemed to be wrong. With the versatility of Replayer's log file handling, we can use the general tools to describe the problem. By using the Google Earth component to display each player's position as recorded on his or her PDA, as well as each player's position as recorded by the server, we are able to see that the discrepancy between these two apparent positions is often very large, and that these discrepancies occur when the players pass out of network coverage. By combining this with an event series showing the times a player uploaded their position to the server, we can see when and why these discrepancies appear: the server maintains the last known position for the player, so when they move out of network range and are no longer updating their position - this becomes increasingly inaccurate, assuming they are still moving, while the local copy stored in the PDAs logs shows the *actual* position (at least according to their GPS log). Alternatively an event series showing a player's signal strength, linked to the original Google earth view can be selected to demonstrate that when a player is connected, the server's accuracy tends to be much higher. Conversely, when a player moves out of network range, the server is no longer updating their view of the others' positions so, again, as time passes, these positions become increasingly inaccurate.

Relatedly, one example case is that of a trial in which the participants claimed in their interview that they had been uploading simultaneously to score extra points. The interviews provide a good starting point for constructing questions that can be answered by the data, especially when viewed in conjunction with the system logs. These players' abnormally low final score brought this into question and, when the data was examined, there were no instances of simultaneous uploads for this team. Further examination of the data showed that one player's PDA was regularly significantly less successful in connecting to the wireless network - thus she was unable to upload simultaneously with her team-mate despite making a valid effort. The visualisation showed she was in an area of good coverage, because the visualisation is an aggregate made up of all the users' connectivity sample data.

What is being highlighted in this particular example is a mismatch between the user's *mental model* of the system and the reality of the *actual* state of the system. Mental model theory [163] assumes that humans create internal representations of objects and situations they encounter in everyday life in order to explain how they work. One important aspect of this is that these representations are 'runnable' in the head, that is, they can be used

to predict the result of a particular interaction with the world. They are not always accurate, which is why humans can have misconceptions about the effects of their interaction. Rehman in particular explores how users may have their mental models of ubiquitous computing systems enhanced by visualisations [178], however in the example described above the visualisations are actually responsible for the breakdown of that model, i.e. the system is showing them something (in this case the apparent signal strength on the map) that may not be accurate. Indeed the network discontinuity on which the game is predicated effectively perturbs the benefits of these visualisations. Kieras [129] explains the learning process for device operation as requiring a good mental model of a system - but here is a system designed to explore what happens when technologies break down - or rather to exploit those breakdowns for some positive purpose. In practice then, it seems that treasure is a game with a difficult learning curve, not because the game or the interface is difficult to learn, but rather because what a player sees does not necessarily map well to reality. Generalising from this finding we can imagine that in a world of ever more pervading location and context aware systems, the discontinuity between what a system might assume (its state) as compared to our actual situation may lead to confusion and the apparent misfiring of systems.

Where systems like Replayer come in here is in helping to highlight the specific breakdowns between users' perceptions of a systems and the systems' perception of users. In this example, if we take either of the data types alone we get an incomplete model. The interviews have the user telling us exactly what they were doing (pressing the upload button when the map showed they were in an area of good coverage). They did indeed receive their points, but fewer than would be expected from a team deliberately making simultaneous uploads. If the system works as expected we can only assume some kind of human error (perhaps they were not pressing the buttons in good enough synchrony). Otherwise we must assume that the system is broken. If we look at the logs alone (without the interview) we see that the users did not appear to upload their data at the same time. If however we use the interview as a cue to investigate this discrepancy and look at the video of the users' behaviour, we see that they were indeed pressing the button at the same time. Because the video is synchronised with the logs we can then see that while one user was able to upload, the other was not (because of differing connectivity on the two devices). The small screen on the PDA makes it impossible on the videos to see what is actually happening - but the logs show us exactly what is happening in system terms. The user is indeed, as they postulated

in the interview, doing exactly what they should, however because the model of the system they have built up by looking at the visualisations does not match with the reality of the system (the device was not in fact connected) their attempt to ‘do the right thing’ failed. This stands as an excellent example of how Replayer (or systems like it) might help us to understand breakdowns in interfaces, and users’ mental models of, ubiquitous computing systems.

### 5.2.2 Pick Pockets

In a related example, interviewees often noted a discrepancy between the number of times they attempted to pickpocket an opponent and the number of times they actually succeeded (and were rewarded with coins). Again, the event series is used to show the occurrences of these events as a starting-point to investigation. Three reasons for this discrepancy present themselves.

The first reason is that the displayed position discrepancy issue described above. It is the *server’s* view of relative player positions that is used to determine whether an opponent is in pick-pocket range, yet players were making visual determinations based on the physical proximity of their intended victim rather than working directly with the information displayed on their PDA. This is an entirely understandable practice. Even without knowledge of the game architecture, the pattern of message events triggered by a pickpocket attempt, as visualised by the event series, can indicate that the success is determined by server positions. The second reason is also clear from the event series: players tended to press the button many times in quick succession when trying to pickpocket, yet because only one attempt could succeed this leads to a very high number of failed attempts reported in the logs, though in this case the interviews suggested that player would only consider this to be a single attempt.

Another possibility, and one that was a deliberate design point of the game, to stop users from just continuously pickpocketing is the shield. When switched on this would protect them from opponents’ attempts to pickpocket. There was however no feedback telling the user that their attempts had been repulsed by a shield, rather the instigator simply didn’t get any coins. The fact that his lack of suitable feedback served to confuse players is a demonstrable flaw in the design of the game.

The final reason is that the intended victim may not have actually been carrying coins at the attempted time of the pickpocket. Figure 5.2 illustrates how Replayer may be used

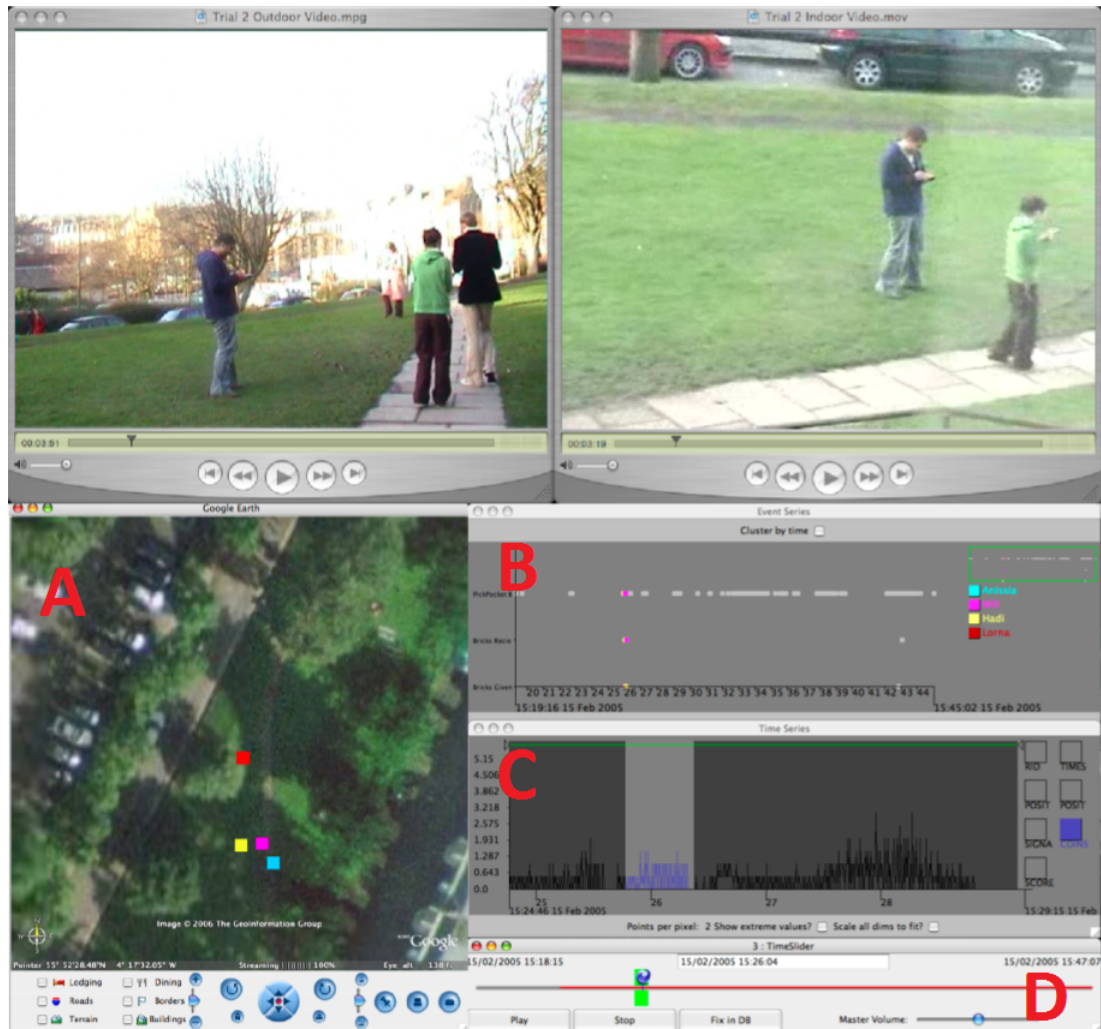


Figure 5.2: Using Replayer to show the context of a pickpocket

to show the context of an individual pickpocket event. The map (marked A) shows the locations of three players involved. The two videos (top) show views of this incident from different locations. The event series (marked B) shows all the pickpocket attempts made in this game, with the current events highlighted. The time series (marked C) shows the number of coins carried by the victim suddenly dropping to zero. Finally the time slider (marked D) shows the position of the incident in the overall timeline.

A pickpocket event is selected in the event series, loading the map with every player's position at this time. A slice of time on the time series is also automatically highlighted by the analyst's selection. From this selected area, the number of coins being carried by the intended victim at the time of the pickpocket can be determined. If at this stage a Media component is open, a selection will highlight the point in the video where the pickpocket attempt was made. In this manner we have selected the area of the videos relevant to the specific area of the data we are trying to understand, supporting the process of thick description.

Locating a pickpocket event simply by examining the video would be impossible, as it is displayed only on the screen of the PDA, and is not an event players would choose to make obvious. Playing the media component when a selection has been made automatically skips between selected periods. By selecting all the pickpocket attempt events in the event series we can see how many of them have been captured on film, and use these films to understand why the player was attempting to make a pick-pocket at this time. Using various components in conjunction we are therefore able to establish through observation for each attempted action, the participant's motivation, whether the action succeeded and, if not, provide a reason for the failure.

In this example we are using Replayer as a tool to support the ethnographic practice of unpacking the character of an event [56]. We are presented with an event which is difficult to understand because of a series of challenges outlined in section 2.3.1. Specifically:

- *Mobility*: our users are out running around literally 'in the field.'
- *Small Displays*: The users are interacting through PDAs and hence the videos are unable to effectively capture the screens
- *Occlusion*: The users occlude each other and each other's screens - and the environment often occludes the users from the view of the videos - particularly in this case where the pick-pocketing user is attempting to hide the fact he is making the attempt from

his proposed victim. This often includes hiding behind trees, or sneaking up on the target etc.

- *Invisible Interaction*: The success or failure of the pickpocket depends on the accuracy of two technologies: GPS and Wifi, quite apart from the games context (i.e. whether the opponent has coins or is using a shield).
- *Distribution of interaction*: The event is taking place between two separate devices using a number of services.
- *Technological breakdowns*: The game is predicated on the discontinuity of network connections as an exercise in *seamful design* [46,47] it is therefore axiomatic that the system will not function in perfect harmony.

In order to unpack this event, we must combine a number of available resources, using the method known as bricolage [64]. We have the interviews in which the users describe exactly what they did in their own terms; we have the system logs which show locations (as understood by the system), signal strength, interface events etc.; and we have the videos which show the *real* locations of the players, and the environmental context in which the event occurred. In order to unpack and describe this complex event we need to make use of all of these resources, and we need them to be synchronised. As example, in the data there are often hundreds of pickpocket button presses - a result what gamers call ‘button mashing’ but few successful events. How do decide which of these is a single event as understood by a player? We can use the event series to show these ‘bursts’ of activity and separate them out into events. Finding the successful ones is easy - but it is generally more interesting to explore the unsuccessful attempts, because these can tell us more about the users’ understanding of the system. In some cases it may be because (as discussed in section 5.2.1) the users’ mental model does not match the state of the system, that is for example, that the system’s stored location for a user may not match their actual physical location - something that only becomes apparent when we look at the ‘real’ location in conjunction with the ‘server’ location. Bearing in mind that the server mediates the event so it is that location which is relevant to the success of the event. Similarly, both parties must be in good network coverage for the event to take place - again as mentioned in section 5.2.1 seeing somebody in a good location on the aggregated map does not guarantee they have a good connection.



It should be plain to see here that there are a very large number of contextual or situational circumstances that effect the success of the event. Unpacking these in detail by making use of the tools provided by Replayer helps us to understand how and why the system succeeds or fails, and how the players are interacting with it and with one another. Once again, we can generalise somewhat from our results and consider that ubiquitous computing is making digital and even physical interaction an increasingly complex proposition to unpack (though it should be noted that this is not a trivial task at the best of times, hence the whole practice of ethnography and ethnomethodology) and we require observational technology to support that unpacking process by making available different views on situations and contexts necessary to understand these new types of interaction.

### 5.2.3 Environment Awareness

Because the games of Treasure took place in the city streets, and it is a fairly intense game, the question arose as to whether users were able to safely operate the system. In one of the interviews a participant stated: “I nearly got run over at one point!” This brought up the question of road safety, as there were several roads intersecting the game area. By selecting the roads in the map component, it is possible to highlight all the points in the videos where participants were crossing the roads. Each event could then be played (if the camera had been focussed on that participant at the time). In fact on examining the videos we were able to establish that players did not, in practice, lose the ability to cross the road or walk around safely. Although they did spend considerable time concentrating on the information provided on the PDA, they also spent much time looking around, for other players, cars and landmarks to help them use the map. This is an interesting observation and worthy of further examination in future work, as it appears to contradict the results of studies like [137]. Is this apparent additional awareness related to the characteristics of this application? - playing with representations of location as it does. Much of the work of playing the game was mentally overlaying the objects presented on the PDA with the physical environment - bringing together the features of the game with the environment they were in. Landmarks on the map had to be read and used to find where coins were and players had to be found for pickpockets. In effect, the players were building a mental model, not, this time, of the system state and interactions, but rather of how the *physical* and *digital* aspects of the ‘game world’ were combined.

If we wish to understand these models - at least in relation to the locations of things the

game knows about (players, coins etc.), it is possible to turn to mathematics. Information theory is a mathematical study of the encoding and communication of information, which provides several measures for the calculation of dependencies and relationships between data sources [194]. One such measure that is of use in the analysis of treasure is *mutual information*; a property used in considering the independence or interdependence of two variables. It measures the amount of information that can be gained about a variable X by knowing about another variable Y. This is applicable to an analysis of treasure as we shall see. As each game takes place in a confined setting, over a relatively short period, it is a reasonable assumption that a player's actions at any time will be affected by the 'state' of the game in terms of other participants' positions and artefacts on the map. Mutual information was measured between the GPS positions of a pair of participants, to assess the degree to which they could be said to operate collaboratively. While this may seem like a radical departure from what has so far been a largely qualitative analysis, we are in fact taking a mixed-methods approach, that is, we will take some quantitative measure (like mutual information) then turn it into an accountable object that can be used to support our qualitative analysis. The nature of system logs generally make quantitative techniques applicable, and it is one of the challenges for Replayer to make the results of those analyses applicable and accessible within a qualitative study.

Such a measure as mutual information(MI) is a more powerful analysis than a simple correlation of positions, as it does not merely detect players moving around in pairs, but more generally attests to the extent to which one participant's location is predictable from a team-mate's. For example, a player might move to one corner of the game space in response to a team-mate heading in the opposite direction, in order to cover more of the playing area, or players might meet regularly in the centre of the map to perform collaborative uploads, before swapping the sides they cover. Such behaviour would be detected by the mutual information metric and would yield high MI scores. Players would only receive low MI ratings if they played completely independently; that is, they acted with no thought to their team-mate's ongoing activity.

Figure 5.3 shows the results from the analysis. The tool on the left is the MI component, which makes a series of MI calculations between pairs of participants. These are passed to the tool on the right, where they are correlated with score showing that teams playing collaboratively perform better. If required, SQL statements to retrieve each of the two streams under analysis are entered in the text fields. A visualisation in the centre of the MI

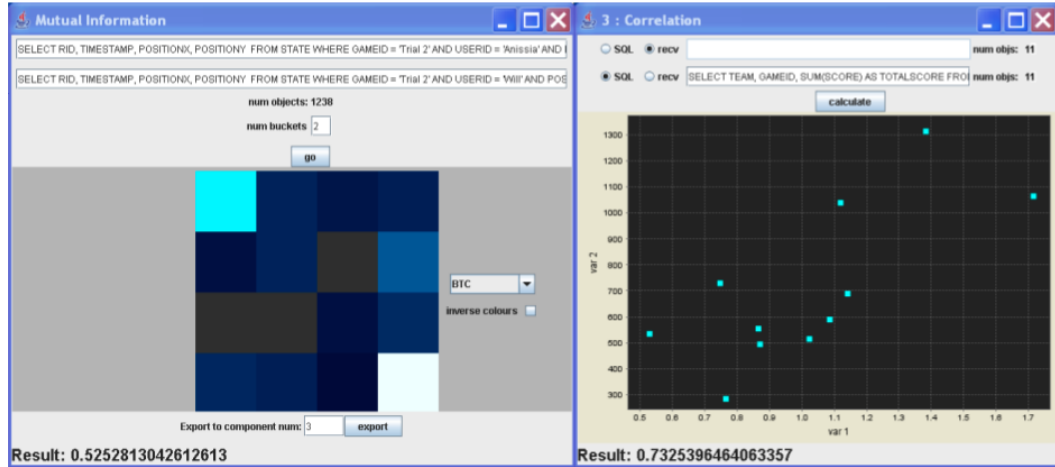


Figure 5.3: Performing a mutual information analysis in Replayer.

frame shows the joint probability density from the current calculation: a plot that may be of interest to analysts, but is unimportant for the current example. The MI result is stored for each calculation - multiple calculations over a series of input streams build up an array of results. The export control at the bottom of the MI tool can then be used to send this MI array to a correlation module, where it can be correlated with other variables.

In this example, data covering six trials of Treasure were analysed, each involving two competing teams of two participants. Therefore twelve mutual information values were generated and passed to the correlation module. The correlation with team score was calculated as 0.527. From the plot, it could be seen that one point was clearly set apart from the others. Highlighting this point in the correlation tool allowed us to drill down to the data from which the point was abstracted, highlighting the associated data in other visualization components. We were able to establish that a system error had caused the logs for this player to be incomplete, thus invalidating both the individual data and the wider MI correlation. This value subsequently removed from the analysis. As shown on the right hand side of Figure 5.3, the correlation of MI and score was then 0.73, a strong positive correlation between MI between team-mates' positions and their final scores. It can therefore be concluded that players acting in a collaborative manner were more successful at the game.

How then is this information useful in context? Beyond being interesting in and of itself, there are a few factors here which require consideration. MI makes no distinction between 'conscious' or 'deliberate' collaboration, and 'subconscious' collaboration. It tells us that teams that seemed to collaborate appear to do better, but if we wish to understand how players interacted with the game and indeed *how* they collaborated then this information

merely serves as a support. It can tell us for example a ranking of the amount of collaboration between teams. We can then use this as a cue to interviewing to try to get a sense of exactly how that collaboration happened. For example, did they actually decide to split up and cover more ground as proposed above? or did they actively walk about together? the MI doesn't tell us this, but there are plenty of resources which can (videos, interviews, maps etc.). Once again we see that to really unpack what happened, in line with qualitative description practices we can take the results of the MI as a resource to support that process. What is interesting about MI is that it will also highlight those 'subconscious' collaborations or interactions (a phenomenon explored in [230]). Without it we would have no good way to explore these in qualitative terms - because we would lack the evidence of their occurrence, but by using this technique of mathematical logfile analysis to create a useful resource, we can explore in depth how this behaviour might affect the experience.

#### 5.2.4 Summary

We have seen in three examples some of the ways in which Replayer was used to unpack interactions. In some cases the data is used essentially as a method of filtration for the videos, as in section 5.2.3 where the video is filtered to find only sections of the video where players are interacting with the system around roads - and the videos could then be subsequently analysed to explore whether players were paying suitable attention to their environment to be safe. Indeed filtration of video is something Replayer provides specific tools for as described in [159] where Replayer is used to automatically find sections in videos containing specific users, and to filter out parts of the video that are of very poor quality. We have also seen, in sections 5.2.1 and 5.2.2 how a combination of resources can be used to understand users' mental models of a system, and indeed how those models may differ from the system's view of a situation. We looked (in section 5.2.2) at how Replayer can be used to tease out the details of an event, allowing a deep understanding and thick description of that event. Finally we looked (in section 5.2.3) at how Replayer might be used to apply mathematical or quantitative techniques to system log data, then make the results of those techniques accessible to inform a qualitative study. We have highlighted how the challenges (as outlined in section 2.3.1) relate to these particular situations and how Replayer can help to overcome them. In each case we have seen how Replayer serves to help a qualitative researcher perform analysis on a complex mobile system,

### 5.3 Day of the Figurines

This case study will describe some experiences with using DRS as a resource to support an ethnographic analysis of a mobile ubiquitous computer game called *Day of the Figurines* [57]. This analysis was done as part of an experience report for the DReSS node. The log files in this case are explored using a bespoke viewer component, added though the log file workbench API and designed specifically for exploring system log data from day of the figurines. Unlike the previous example, this study represents a *real world* case of the use of DRS. DReSS has the advantage of a strong user base for testing within its own staff, due to its participatory design predicated driver project process, with examples of users including ethnographers, psychologists, and English linguists. This has given DRS a strong development cycle opportunity with testing taking place in the field by users other than its core developers, and rapid prototyping of new tools and processes.

The development of DRS supports the process of ethnographic description and has added value to the core business of ethnography, i.e., the writing of culture in details of the practical action and practical reasoning of its members. We start from the beginning with system logs to articulate systems' support for ethnographic studies of ubiquitous computing. Below (Figure 5.4) is a representation generated from a system log of a cultural experience created by Blast Theory<sup>1</sup> and the Mixed Reality Lab from Nottingham University. Day Of The Figurines is a mobile SMS based game set in an imaginary city played by hundreds of geographically dispersed people over four weeks.

The representation parses the system log and thus transforms the raw log into a human-readable and accountable object; that is an object that may be discussed and reasoned about and which can be drawn upon to formulate accounts of game play. Regardless of whatever technical characteristics 'parsing' might have, the term refers to the co-design of representations that address specific ethnographic interests. Co-design means that the representation is the product of collaboration between the designer and the ethnographer. The purpose of the collaboration is to specify features of the log that are relevance to ethnographic research.

Practically, there is a reason here why a generalised component like Replayer's mapping tool is not feasible for such a project. The game is predicated on the existence of a 'virtual world' Though in practice there *is* a physical representation of this world in the form of a model at a museum on which players place their *figurines* at the beginning of the game. The

---

<sup>1</sup>[www.blasttheory.co.uk](http://www.blasttheory.co.uk)



Figure 5.4: An XML based log from Day of the figurines

state of the game is maintained by game organisers physically moving the figurines around the map as information comes in. Representing this board might be considered in a number of ways - not least by simply filming, or taking regular pictures of the board. However, we must consider the meaning of location in this context. There are two locations of interest:

- The *real* location of the player - which is not tracked by the system, and which, though it may be relevant to how the player interacts with the game, is not strictly speaking part of the game's 'state.'
- The location on the board of the player's avatar. It is also important to realise here that this is not a cartesian coordinate relative to the board - and thus easily overlaid on a photograph of the board, but rather an abstract location related to landmarks like 'school,' 'hospital,' 'fire station' etc.

As such the snesible approach appeared to be to create a project specific mapping component - essentially a representation of the board which could be used with the relational database generated from the log files to determine *who* was *where*, and *when*.

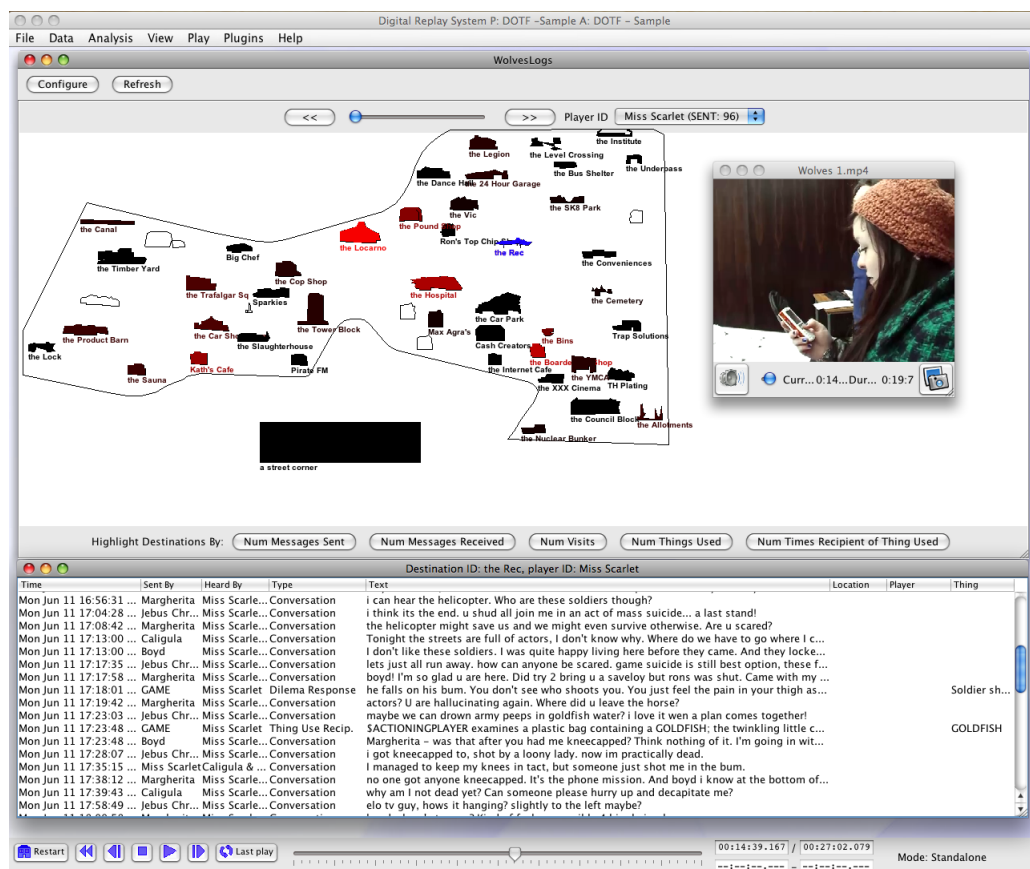


Figure 5.5: Seeing what players did at locations.

Thus we have a virtual model of a physical model of the imaginary city that all the players saw when they registered to play the game at Wolverhampton Art Gallery. The virtual model shows all the locations in the imaginary city that players could visit. More than that it allows the researcher to query the log by specific ‘player ID’ (top right Figure 5.5). The *player ID* menu shows all players in the game and the number of messages they sent (e.g., Miss Scarlet sent 96 messages in Figure 5.5). We can then highlight *destinations* or *locations* that a player visited. We can do so by *number of messages sent* in a location, *number of messages received* in a location, *number of visits* to a location, *number of things used* in a location, and *number of times recipient of thing used* or gifts received from others in a location (see bottom of Figure 5.5). Effectively the interrelatedness of the database combined with the visual components creates a visually queryable resource, that is, one that can easily answer questions about who was doing what, where and when from log data stored in a database, and the data is returned in a usable fashion.

For example, when *number of messages sent* is selected, each location a message was sent from is highlighted. Blue indicates where the player is on the current timeline, in this case the Rec (as in Figure 5.5); red indicates where most messages were sent from; the tone of red decreases across the remaining locations according to messages sent. The same principles apply across the other methods of highlighting locations visited by a player. The slider (on the left of *player ID* in Figure 5.5) allows us to trace the player’s route from location to location over the four weeks of the experience according to the method of highlighting selected. Thus we can scroll through the player’s virtual journey to see at-a-glance the locations they sent messages from, received messages, what places they visited and how often, what things they used in those locations and received from others. We can also see at-a-glance the people they talked with at any location. To see just what was said and went on between players any highlighted location we can click on the location to reveal specific interactions.

Drilling down into a location gives us a view of interaction involving the selected player (e.g. Miss Scarlet) over immediately previous, current, and following days in details of *time*, who the message was *sent* by (including game events from the server), who the message was *heard* by (or at least received by), what *type* of message was sent (e.g., a conversation, a message describing the thing a player has requested to use, a local game event, and so on), the *content* of the message, and the *thing* requested by a player. Message *location* or *player* who sent the message are also available according to type, specifically when a player



arrives or requests to leave a location. Drilling down into a location provides us not only with a view of current player interactions at, for example the Rec, as in Figure 5.5, but with a detailed retrospective-prospective view further articulating the context of interaction in that location. Furthermore, we can click on entries in the ‘player’ list and see the locations they arrived from and departed to from the current location and see the details of their interactions in those places.

What we are looking at is a reconstruction of events. Beacuse of the casual and slow nature of the game (which is played over four weeks) and the large number of players, a more traditional approach to capturing this data for qualitative analysis, such as video is frankly infeasible. While video can be (and was) captured for short periods for a subset of the users, getting an overview of ‘what happened in the game’ can only be reasonably actioned by use of system logs, and given the complexity of the events a custom viewer was necessary. This incidentally serves to show why extensibility is crucial in systems like Replayer and DRS. No matter how many general purpose tools we provide, there will always be situations and experiences that those tools are unable to adequately explore. By making it fairly trivial to add those tools - at least to a programmer who need not have a depth knowledge of the system he is building a tool to analyse, but rather just the requirements of the tool, we support a much wider potential user community. So the board representation behaves as a queryable qualitative resource, unlike any of the more familiar tools (video, audio, field notes etc.) but powerful and reasonably easy to use in its own right. It also maintains its state when interacting with the document viewer (3.3.9) so states can be saved and revisited as notes in the wider context of a description.

At the click of a button, rather than after a prolonged manual effort, DRS enables us to extract relevant sequences of interaction from system logs; or rather, at the click of a button after the co-design of appropriate representations that transform the log into an accountable object. The production of representations from system logs is very much bespoke at the moment and requires close cooperation between designers and social scientists.

While there will always be a bespoke element insofar as the study of novel computing domains and applications is concerned, one of the key challenges for the future development of DRS is the production of standard representations or *viewers*, such as those offered by Replayer, that enable social scientists to interrogate the contents of system logs. Nevertheless, Day Of The Figurines provides a view on system logs that makes the temporal and spatially distributed character of social interaction and communication visible, it enables

the analyst to trace players' routes and journeys through space (in this case the virtual space of the game), and it allows us to inspect collaboration between players in detail. Had the players' phones been equipped with locational technology their *real* physical locations could also have been modelled allowing an exploration of the relationships between physical location and interaction with the game. While we could imagine refining and extending the interrogative functions of the viewer, providing better graphics or viewing multiple players' pathways and their intersections for example, the need to develop viewers that provide spatial and temporal views is key to the application of DRS to interaction, communication, and collaboration in the digital society at large.

Of course this is only one view on the system, used to get an understanding of exactly what happened in the game. However, if we wish to understand the experiences and interactions with the game from the perspective of a user, rather than simply the system, we must capture one person's experience more thoroughly. This is done, using the more familiar approach of video ethnography, as described in [102]. Analysing the data from the perspective of this video alone is again subject to the challenges described in section 2.3.1. In this case:

- *Mobility*: Discussion with players suggested that much of the interaction with the game happened in 'off moments' such as travelling or standing in queues. While they are not exactly running around in the way that players did in treasure or other Blast Theory games like *can you see me now?* and *uncle roy all around you*, the fact remains that the game takes place (at least from the perspective of the player) on a mobile phone.
- *Small displays* As above, the game takes place mobile phone via SMS. This is not a format that lends itself well to video - indeed much effort is expended in dramatic storytelling to figure out how to embed this particular quiet, but extremely popular communication channel in broadcast media - i.e. a character gets a text, how to show the contents without them reading it out loud, which would be unusual and potentially story breaking, or creating jarring graphics.
- *Non-collocation*: The player is geographically separated from the game board, and usually from all or most of the other players.
- *Distribution of interaction*: There are many players interacting through mobile phones, but on the other end of that there are organisers with computers and a physical board

to manage.

- *Interaction Time*: SMS is not a synchronous medium. The time it was sent and even received does not necessarily map to the time it was read or the time it was acted on. In purely system terms what matters is when and what the actual messages sent are, but if we want to understand the ‘game’ and how the players interact with it we need to understand this.
- *Technological breakdowns*: The game depends on mobile phones. These are not always on, and not always in range. How might this affect peoples’ interaction with the game? If a player turns their phone off while at work - as was discovered to be the case in one example, and misses a crucial game event, what happens? how does this affect their experience?

In practice however we can address much of these by simply combining our videos with the representation discussed above (as per figure 5.5). We do not need to see the specific messages on the video because we have them right in front of us and synchronised on the viewer. We also have to had the game context in which that event took place - who sent it? from where (on the game board)?, why? etc. We can address these questions generally with the system log data, and of course it would be feasible to do this just with a list of paper logs, using the process discussed in section 2.6, but the time and effort involved would be extremely significant. DRS really does add value to the whole process by giving us context, and allowing us to ask questions of the data in a usable manner. It creates an accountable resource through representation that can be used with other resources to understand and describe interaction.

Analysis of interaction, communication, and collaboration is not restricted to what can be seen via the viewer. While it obviously provides a resource that may be used to analyse the inner life of the game, or of digital interaction, communication, and collaboration more generally, it also supports exploration of the game’s intersection with the everyday life of participants as explored in [21]. Digital interactions, communications and collaborations are embedded in physically situated interactions, communications and collaborations. Thus, we may be in the physical presence of people and also be interacting with others at a digital remove. Understanding not only what people do in digital environments but how we weave digital technologies into our physical relationships - where we use the digital, when we use it, how we use it and so on - are all key issues to understanding the impact of

ubiquitous computing on everyday life and arguably to broader social science research into the digital society as well. The kinds of tools described here give qualitative social scientists the resources and opportunities to study those relationships. Spatial and temporal views on digital interaction, communication and collaboration provide a resource to explore such themes, providing a concrete and detailed starting point for conducting rich interviews to complement our understanding of what goes on within digital environments with external dependencies that shape digital use.

Just by attending to ordinary problems of extraction occasioned by the use of system logs in DRS we can see that DRS does not just make extraction easier, it transforms the extraction process from one that is concerned to make log contents intelligible into one that adds value by enabling interrogation and inspection. Instead of labouring through logs seeking out relevant sequences of interaction, extracting them and cleaning them up, the work is replaced by the co-design of log viewers to represent salient features of interaction, communication and collaboration. With the development of standard views much that work may be dispensed with too and the analyst may concentrate on what system logs have to say as it were, rather than on making them into accountable objects.

### 5.3.1 Summary

We have seen how DRS enables the analyst to examine not only the contents of system logs but other resources too in coordination with those logs. This particular case study is of interest because the primary feature discussed is a bespoke viewer created for analysing the Day of the Figurines project from within the DRS environment. A new representation was created within the system, developed in close collaboration with the users to support exactly the necessary forms of data and data interrogation. We have shown how a representation of the data can be used to understand state over long periods of time - as was the case with day of the figurines, and how that can be used to provide context to shorter in-depth studies of specific parts of the process through video ethnography or other means. We have seen specifically how DRS can be used to overcome the challenges outlined in 2.3.1 and presented by day of the figurines, and how it has been used in a real study by a professional ethnographer.

## 5.4 Other examples of the use of Replayer and DRS

Aside from Treasure, Replayer has also been used in the studies of *Feeding Yoshi* [21], where system log data was used to psuedo-spatial maps of player locations, and *Shakra* [9] where it was used during the iterative design cycle to demonstrate the success rate of the machine learning algorithm behind that system.

DRS, having been a significantly longer running program has a wider array of users. It has been used in the construction and coordination and coding of of the Nottingham Multi Modal Corpus [42], It has been used in the study of VIRILE [35] where it was used as a tool to code and synchronize several simultaneous videos, system logs and a screen capture, It has been used by the Real Life Methods group to synchronize several different video streams capturing a group session, and to link the data returned in questionnaires to related parts of the discussion. It has been used by the thrill laboratory [188] to support public playback of biological monitor data (heart rate, galvanic skin response etc.) It is also in use at Nottingham University’s psychology department and in particular the *visual perception and driving in autism group*, where it has been used as both a video playback and interaction data capture device as well as a coordinated playback device for eye-tracker data along with the original videos being watched at the time. DRS is currently in us in a number of other studies involving several different universities and departments. As of January 2009, DRS had been downloaded by 538 distinct IP addresses 423 of which were external to the University of Nottingham. All users of the system register their usage online, and later this year a survey will be conducted to learn how it is being used and on what type of data.

## 5.5 Conclusions

We have explored through these case studies some specific evidence of how Replayer and DRS can add significant value to an analysis. With reference to the challenges laid out in section 2.3.1, we have seen how each of the case studies presents a subset of those challenges and how Replayer and DRS have been used to overcome them. We looked at ways in which we can use multiple resources in *bricolage* to unpack interactions, particularly in cases where parts of that interaction are *invisible*, or *uncapturable* (5.2 and 5.3). We looked at how we can inform our analysis in terms of what to look for and questions to ask in interviews by applying quantitative techniques like mutual information to our system log data (5.2.3),

and how we can use system logs, in coordination with other resources to better understand users' *mental models* of systems (5.2.1 and 5.2) and how inconsistencies between those mental models and the system's internal model may cause interaction to break down.

We have also seen that in some cases, general tools are going to be insufficient, and that these systems need to be sufficiently extensible to allow the addition of new representations of the data. Here we see, as has been discussed, the intrinsic need for a person with skills and knowledge of both computing and social science practices to provide much needed communication between the social and computing science communities. We also see the benefit available to a willing social scientist using the tools provided by computing science, and it is this collaboration which will serve to continue to drive forward the use of social science as an analysis tool for ubiquitous computing in general.

It is important to take away from these studies that they are very much anecdotal. They can never be true formal experimental evaluations of the tools for the very fact that doing such an evaluation is infeasible. Because of the inherently reflexive nature of ethnography, it would be impossible to design an experimental scenario where the same system could be analysed with and without the tools and compare the results. Instead we must rely on the evidence of examples, feedback and the fact that the tools (in particular DRS) have been designed iteratively and in close collaboration with a number of professional social scientists. Certain functionality, for example, the *document viewer* (3.3.9), very specifically supports ethnographic process as laboriously determined in interviews, discussions and testing. Ultimately these case studies can only scratch the surface of the functionality, or analytic possibilities to be found in either system. Much of DRS, for example, is concerned with corpus linguistics, as discussed in depth in sections (3.3.10 and 3.3.11) as a result of one of the driver projects, and these tools have not been included in the case studies. So while these systems have not undergone a formal evaluation, they have been extensively "tested" through their use. DRS in particular has been prototyped, tried out, modified, tried again and generally gone through an iterative design process with often several prototypes of each tool (at least for the more complex ones) each building on feedback from its use in one of the driver projects. A good example of this might be the coding tools, used and tested extensively in several projects including the Nottingham Multi Modal Corpus [42], and VIRILE [35] both of which were undertaken by members of the driver projects. It was the feedback from those users which has helped DRS's coding tools to be effectively usable.

It has been the intention in this chapter to show that there is a genuine practical use

and value to be had when doing qualitative analysis from these types of tools and this kind of coordination of data.

## Chapter 6

# Reflections and Conclusions

### 6.1 Replayer vs DRS

In the above chapters we have explored two very different pieces of software, Replayer and Digital Replay System (DRS). This section will examine the different approaches taken, including the reasoning behind those different approaches, and explore the strengths and weaknesses of each.

The goal of each system is ultimately the same: to combine heterogeneous media, including system log files onto a synchronised, coordinated view in order to support the practice of qualitative data analysis in a world where the miniaturisation and pervasiveness of technology makes system logs that an increasingly important and useful resource.

Replayer approaches that goal very much from the side of the system logs, providing a variety of generic ‘chart’ type viewers that can be used singly or in a coordinated fashion to explore that data. It takes an approach familiar in the world of information visualisation, using brushing and linking between different views to allow filtration, selection and exploration of that data. Replayer’s rigorous organisation of those logs allow this system to function effectively, while the distributed nature of its architecture allows it to circumvent some of the major issues generated by a multi-view system such as screen real-estate, lack of processing power etc. That same architecture could also allow multiple analysts to work together concurrently, or analysts to work directly with a system’s designers to further a better understanding of the data contained in the system logs. To provide playback of other types of time based media, such as video or audio, Replayer makes use of tools familiar to most users: QuickTime on OSX and windows media player on Windows.



Key strengths:

- Cross Platform support
- Well defined internal database structure for storing logs, based on a sound framework
- Large range of viewer types for those logs
- Instrumentation system to support the logging process
- Brushing and Linking based synchronisation of data, allowing complex selection and filtration processes
- Support for collaborative analysis and/or separating complex processing tasks to different computers
- Strong framework for extensibility in every aspect
- Uses familiar tools for playback of familiar media types (QuickTime, Windows Media Player)

Key Weaknesses:

- Unfamiliar approach for social science
- Lacks the more ‘expected’ CAQDAS tools such as transcription, annotation and coding
- Lack of flexibility in log input
- Weak synchronisation for media files
- Weak playback synchronisation
- No support for corpus management
- Knowledge of SQL required to construct more complex queries
- Not a very intuitive interface
- No support for project meta data
- No real support for project/data sharing
- Lack of ways to ‘search’ the data
- No support for images

It is important to consider that Replayer exists really as a proof of concept. It serves to demonstrate the kind of things that *can* be done by combining system logs with other media data as shown in the case study. However, its complicated interface, dependency on SQL and lack of support for other CAQDAS methods make it not really suitable as a ‘real world’ tool by social scientists who do not also have a strong background in computing science. The environment in which it was constructed was largely one of computer science rather than social science.

Conversely DRS was constructed to actually be a tool usable by social scientists. Taking a participatory design approach the development of DRS was guided at every stage by three main *driver* projects comprising different aspects of qualitative social scientists:

- Ethnographic analysis of ubiquitous computing technologies
- Linguistic analysis of multi-modal features of natural language use
- Psychological analysis of teaching and learning in e-Learning environments

The varied backgrounds in qualitative social science of the three driver projects has allowed the development of DRS to focus on supporting those varied practices culminating in a tool more widely applicable than for simply one area. As an example, consider the different coding systems within DRS. The time-based coding in the track viewer is something required by both the English linguists and the psychologists though the particular ethnographers who comprise that driver project have little interest in it. Conversely the coding (technically ‘markup’) provided in the DRS document viewer is designed to directly support the process of constructing ethnographic reports and was designed in close collaboration with those ethnographers.

It is perhaps unsurprising then that DRS has the look and feel of a familiar CAQDAS toolkit such as ANVIL or Transana, though it provides an amalgamation of the facilities of several different established tools. DRS is much more rigorously engineered than Replayer, since part of the project’s mandate was to create a tool that would be freely downloadable for use within the social science community. Because of the particular focus of the driver projects, log files actually had a lower priority in the early stages of development, despite being the feature that truly sets the system apart from those other CAQDAS tools, however support for them was designed and built in from the earliest stages of the design, and this is reflected in particular in the way meta data can be applied to log file data in the same way as other data types, and in the way the log file workbench has been designed to handle

the input of many standardised logging formats. DRS has however been criticised by users for its inherent complexity. Being a ‘monolithic’ tool, all the functionality is exposed all the time, giving it a very steep learning curve for a new user.

However, despite some criticism, DRS has been picked up and made use of by a number of social scientists, working in different fields and doing ‘real’ work. Section 5.4 highlighted a number of projects making use of the system, as well as a large number of downloads (423 outside Nottingham University). It is directly supported by the CAQDAS networking project at the University of Surrey, an independent group who provide training to social scientists in the use of software tools <sup>1</sup>. It is difficult to refute the fact that it is being used in anger, and has produced real results, such as those reported in [42, 96, 140, 219]. The general feedback has been that it is not completely straightforward to use, would benefit from some improvements to its interface and some debugging, but is usable and provides an effective method for approaching data.

One interesting example of this usage - or more properly of appropriation, was in the Riders have spoken project, described in [96], wherein DRS was used not so much as an analysis tool, but as a method of archiving large amounts of experimental data. The project and analysis system it provides, along with its support for a wide variety of data types apparently made it very suitable as a data curation system, which just happened to have some added benefits for marking up that data.

Key strengths:

- Cross platform support
- Familiar tools for qualitative analysis (transcription, coding, annotation etc)
- Support for many different log types, timestamps etc
- Well engineered for a stable, consistent and familiar interface
- Strong support for Meta data
- Strong corpus management tools
- Support for project sharing
- Included tools are designed to the requirements laid down by social scientists
- Support for import/export to and from other tools such as Transana and SPSS

---

<sup>1</sup><http://www.surrey.ac.uk/sociology/research/researchcentres/caqdas/>

- Cross media searching
- Supports a wide range of data types: logs, video, audio, pictures, transcripts, documents
- extensible ontology able to handle additional data types as they are included
- API for creating custom viewers

Key Weaknesses:

- Very limited tools for viewing/analysing logged data
- Number of possible concurrent views limited by screen size and processing capability
- No support for the actual process of logging
- No standardised internal database framework for handling logs
- Weak synchronisation for media files
- Weak playback synchronisation
- Difficult learning curve

If we compare the lists of strengths and weaknesses of the two systems it is interesting to note that in many cases, the strengths of one, are the weaknesses of the other, and vice versa. The reality is that a combination of the facilities provided by these two tools is needed to really provide the kind of tool that should be possible. The two weaknesses shared by both systems (that of *maintaining* temporal synchronisation during playback, and getting a sufficiently high-accuracy synchronisation to start from) has been heavily addressed in DRS's sister project MiMEG [82]. Were that functionality also to be incorporated those weaknesses could also be addressed. As should be expected, DRS being the larger of the two projects provides a more usable tool, but Replayer is arguably the more innovative of the two focussing as it does on what can be done with the system log data. However, visualising log data on its own is nothing new. Replayer began the process of integration with more traditional materials by supporting the synchronisation of video and audio files with that log data, but DRS provides far more support for the use of those types of data.

### Extensibility

Let us now look at a one area in which both systems excel: Both Replayer and DRS are very extensible in terms of creating new visualisation tools. DRS also allows for the creation of new types of metadata by the expansion of its ontology. In Replayer's case, new tools need only be able to communicate by raw sockets and understand a fairly simple string-based protocol to be able to become part of the system. For them to appear in the interface they may need a java wrapper round any other language software (such as an executable file), but that is a trivial process. There are essentially two areas for expansion in Replayer: parsers and visualisations and both are quite simple to create. DRS similarly provides opportunities to create parsers - called processors in DRS but doing basically the same task of transforming log files into database tables) and visualisations - called viewers in DRS. Unlike with Replayer, we have a strong example from the case study where a viewer was made for a specific project, and indeed another example from [96] where a 'map' viewer was created for DRS to show the paths riders had taken. This extensibility makes the two systems extremely flexible and is one of the key aspects of the software that makes the practically useful for dealing with ever more complex interactions and experiences. It would never be possible to create a tool able to handle every situation, but to create a tool able to handle lots of situations and put infrastructure in place to allow others to add to that tool is practicable.

## 6.2 Revisiting the challenges

In the related work section a series of challenges was laid out defining some of the difficulties associated with trying to perform qualitative analysis on ubiquitous computing systems, and more generally just in an environment where ubiquitous computing is an increasingly normal part of everyday life. In this section we will revisit each of those challenges and describe how the software explored in the previous three chapters attempt to manage them, with specific reference in each case to the case studies outlined in the previous chapter.

### 6.2.1 Mobility

This is certainly a key factor in virtually all ubiquitous computing systems. Users of ubiquitous systems are often mobile. They move across extended physical areas, quickly at times, sometimes even running, which can make the documentation of action and capturing

of video material difficult at best. However analysts are able to leverage ubiquitous computing technology to support the analysis process. For example if the study participants were equipped with global positioning systems (GPS) and those GPS devices recorded a log, then a viewing component such as Replayer's *Google earth bridge* Would allow those location traces to be plotted on a map. Recording additional system log information about their interaction with the technology around them allows us to replay exactly what they were doing from a purely *systemic* point of view. Combining that systemic information with video, possibly captured from several sources and synchronized with one of the replay tools, and further combining that with the field notes and knowledge of our analyst it becomes easier to build up a flexible corpus of useful information that can be used to effectively describe the character of the user's behaviour.

Turning to the case studies, we are well served here by the examples from *Treasure* (section 5.2), which is a fast paced game played out in the streets with PDA interfaces. When performing an ethnography using more traditional observation techniques such as video, we are unable to capture the 'hidden' game events and states. It is difficult to attend to all the players at once, but as they are interacting with one another, sometimes at a distance through 'pickpockets' (5.2), simultaneous uploading (5.2.1) etc. we need to have a deep understanding of the *context* of the game, i.e. its state to make sense of these interactions. A set of recorded positions for each player (both from the server and the client - the discrepancy of which was shown in sections 5.2.1 and 5.2 to cause players' mental models of the system to break down) allows us to reconstruct the movements of each player after the event, and give each the attention it deserves. We also saw that by applying mutual information theory to the respective players' movements (section 5.2.3, the system was able to inform us about collaboration on both a conscious (as shown by discussion in interviews and recorded on game videos) and more subconscious level.

Similarly with *Day of the Figurines* (5.3) the players often play the game in free time such as while commuting or waiting in queues. The fact it occurs via SMS on mobile phones makes it almost axiomatically a mobile experience. Capturing data from these events is challenging in this case also because of the time factor. The game is played over four weeks, with mobile participants, making a more regular observation difficult. By creating a 'view' on the game constructed from system logs, we can examine the general behaviour in depth and when we do gather observational data about short periods of play we can understand this in the context of the game itself and the surrounding players.

### 6.2.2 Small Displays

Interaction frequently involves the use of devices with physically small displays such as handheld computers and mobile phones. This makes it difficult for an analyst to see a user's interactions with a system. Of course there are many ways to capture that information, through long established human computer interaction research. Having users 'think aloud' their interaction springs to mind, but methods like that rather break the 'real world settings' exploration favoured by many analysts. Again then we may turn to replay to provide a solution. One option available for some devices is to use a screen capture approach - i.e. record a continuous video of the screen using special screen capture software. Again replay software such as DRS or Replayer can be used to synchronise this video with other videos capturing a wider view of the context of the interaction. This approach has successfully been used in an evaluation of the VIRILE system [35] and [34]. Alternatively a well instrumented system such as Treasure may have much of its action reconstructed from the system logs as demonstrated in the case study. This reconstruction may then be used as a resource in synchrony with other resources such as video, field notes etc in the production of an analysis.

Turning once more to the case studies, we can look at the practice of reconstruction. In treasure (section 5.2), the screen gives a lot of information about state, but we are unable to capture this (screen recording is not technically feasible on PDAs) because of the size of the screens. For a good example, we might turn to the case of reconstructing a pickpocket event (section 5.2). In this case capturing the screen is made even harder by the fact that one player is trying to hide his actions from his target. There are a number of circumstances which might cause a pickpocket to succeed or fail, and we can reconstruct those circumstances - both in terms of what information the player had (leading to his attempt) and what the state of the system was (leading to the result). In Day of the Figurines (section 5.3) The interaction takes place through text messages. If we are observing natural behaviour, people do not generally read their text messages out loud. However our synchronised reconstruction of the game events allows us to see those messages (i.e. what the player sees) in the correct context and thus better describe and interpret their actions.

### 6.2.3 Headphones

Users often have audio information provided through headphones which becomes unavailable with traditional capture techniques like video recording. This means that the analyst may

miss much key information. By simply recording the audio sent over the headphones and playing that back in synchrony with other data using the tools provided, we can recapture the crucial information that might otherwise have been lost to the analyst using more traditional capture and playback techniques.

In this example, the case studies cannot help us, because neither of the examples we explored featured audio or headphones. However the same holds true for this as for the previous example. If we have the audio recorded separately and synchronised, or it can be reconstructed from system events (as is the case with some forms of audio feedback) We can overlay that audio track with captured video - or other form of observation to better understand the context and thus better unpack the behaviour.

#### 6.2.4 Non-Collocation

When interacting with collaborative ubiquitous systems users are frequently interacting with other users who are not collocated with the user. This characteristic of ubiquitous computing systems presents a significant challenge for an analyst. One possibility is to capture both sides of an interaction using either system logs or other recording technology such as audio or video then reconstruct the action post-hoc using one of the replay tools described. Additional information may be recovered by interviewing the participants then using tools provided to annotate the recordings, with information regarding the other side of the interaction. Again we have a situation where the tools allow us to reconstruct a view over the data which would have been difficult to achieve without the support provided by these tools.

The case study that serves us best for this point is Day of the Figurines (section 5.3). Players are geographically separated from each other, from the game board, and from the organisers, and communicating with the game through a non synchronous medium (namely SMS) - which means they may also be temporally separated e.g. receiving events at different times - something that will be discussed presently. The geographically dispersed nature of the game makes it challenging to observe interaction between players, but we *can* observe one side of that interaction and reconstruct the missing information (the other side of the interaction) later when we unpack it.

The study of Treasure (section 5.2) does also serve here in the same manner. While the distances are smaller, since treasure was played over an area of only around one square kilometre, they are still widely dispersed over that area. As such attending to all of them at



once is difficult, so the same reconstruction of context serves us. One related and interesting factor coming from the case study is that of ‘collaboration at a distance,’ something suggested by the mutual information analysis in section 5.2.3. It showed us that players were able to divide up the space in a form of collaboration that might be difficult to capture or indeed even be aware of when they are some distance apart.

### 6.2.5 Invisible Interaction

Users often interact with invisible sensor systems such as Global Positioning Systems or video tracking, which can make it challenging to understand why users are acting in a given way and how the sensing systems are actually behaving. The inclusion of synchronised log data from these systems allows the analyst to include this missing information in his analysis, and thus better describe the accountable process of an interaction.

The case study of treasure (section 5.2) serves us well here. In the specific example in which we wish to understand how players failed to successfully upload coins in a particular location (section 5.2.1), it is only apparent what is going on if we can reconstruct exactly what data was available to that player. Looking at their position we can see that they are in what appears, at least on the aggregate signal map, to be an area of good signal coverage, and by combining this with information (also shown on the screen) which shows they were not in fact connected to the network, we are able to understand why the upload was not succeeding. The players did not notice at the time that this was not working, and there is no indication on the video, though it is discussed in the interview when talking about their seemingly low team score and their attempts to collaborate. The interview highlights a problem that cannot easily be answered through observation - but reconstructing the context gives us that answer - pointing again to inconsistency in mapping between mental model and system state.

### 6.2.6 Distribution of Interaction

Interaction may be distributed across different applications and devices. Interaction is thus not only located in different physical locations but may also be mediated through different applications and devices, which makes it difficult to develop a coherent description of interaction. Again the support for synchronised system logs helps to support the collection of interaction fragments into a cohesive whole. Bringing all the logs into one playback creates a holistic ‘virtual device’ which combines data from all the devices and applications into a

synchronized corpus for playback and analysis. It is thus easier to consider the system as a whole rather than what it is in reality: a collection of diverse technologies and applications. This ‘virtual device’ can then be examined in synchrony with other materials, allowing the analyst to focus on the character of the interaction, while remaining aware of the diversity of resources supporting the description.

Both case studies give us examples of this issue to explore. Treasure (section 5.2) is made up of a series of PDAs and a central server. To get an accurate picture of the ‘state’ of the system, logs are recorded on each of these devices, then combined together to create what is, to all intents and purposes, a ‘view from nowhere.’ However it is through this view that we are able to reconstruct the context of and subsequently unpack events, as in the example of exploring a ‘pickpocket’ event (section 5.2). In that event the user looks at his PDA interface, which gives him a picture of the known game context - or at least some of it (recall that he is not informed if his target is using a shield, nor if his target is out of network coverage). he then makes a decision to try to pickpocket a target. When he clicks the button, a message is sent to the server, which examines its own view of the context, measures the distance to the target and accepts the event only if its position for both the player and target are within range of one another. A message is then sent to the target’s device telling it to remove coins and if the receipt of that message is acknowledged (confirmation that the target was indeed in range) a message is then sent to the player’s device telling it to add the coins (i.e. that the event has succeeded). As we can see here, even without the shield or other factors, there is a complex process to go through involving messaging from several systems and a dependency on two technologies (GPS and Wifi). Only if all this succeeds will the player have made a successful pickpocket and to understand why one may have worked while others may not have we must build a picture of the system’s state, and the player’s mental model of that state. If the two do not match then the attempt will fail and the player will be disappointed. We see then what the benefit of being able to reconstruct all this information is - and combining this with videos of the event helps to shape the understanding of the process. Some players for example simply mashed the pickpocket button whenever they could see another player. Others were more deliberate - checking their screen to make sure everything was in place, or lying in wait near the upload areas. These actions are best unpacked through the coordination of video, system log data (suitably visualised) and interviews. The study of Day of the Figurines (section 5.3) serves us equally well. That system is made up of consumer mobile phones, an SMS server, a game

server and of course the physical board. A player-player interaction is not a direct SMS but one mediated by the servers, which record the event. In fact the case study is a great example of the representation of a system as a virtual device. We are able to construct a tool for querying the event data (as seen in the case study and pictured in figure 5.5, regardless of what types of phones people might be using, what network they might be on etc. and we can use this representation to unpack players' interaction with the game in considerable depth - something that would have been extremely difficult without that representation.

### 6.2.7 Interaction Time

Certain communication channels such as SMS messages or email are not continuous, that is a message may come in at one time, but a user may examine it much later then wait till a convenient time to respond. This complicates the process of conversational analysis as the context of the times when a user receives a message and when they act on it may be very different. Synchronised access to all the resources involved allows the analyst to select exactly the resources he needs to develop his description, filtering out those which are at the time irrelevant. The facility provided by Replayer and DRS to jump straight to the relevant areas of a video tied to a specific system log event means that the analyst can easily focus on one particular interaction, even if the parts of that interaction are not temporally collocated, and construct a cohesive description of that interaction.

The case study of Day of the Figurines 5.3 provides an excellent example of this phenomenon occurring. The game is played over SMS, which is an asynchronous medium. The server records when it sends a message, and when it receives one, but cannot know when that message was received by the player (the phones are not instrumented) nor when it was read. Because the server runs continuously, we can have some idea of when the message was acted on - assuming good network coverage ensures the reply is delivered promptly. This is an example of why dealing with the system logs alone is not sufficient to build up a picture of how people engage with the game. We know how *in game terms* they interact, but not the context under which that interaction took place, or the thought and decision making processes behind that interaction. For this we must turn to more familiar methods: observing and recording the player in situ for example, or through post hoc interviews. We can however inform such interviews by examining the data in the viewer for interesting game events to explore with the players - and indeed because we have a timestamp for the message events we can interrogate the player as to why particular delays took place. In the

case study, for example, we have the case of a player who switched the phone off while at work. We need a combination of the temporal and system context provided by the viewer, with the observation or interview based understanding of the players actions in that context to fully unpack what happens in their interaction with and through the game.

### 6.2.8 Technological Breakdowns

Because ubiquitous computing systems frequently rely on the use of diverse technologies all being used together, sometimes not everything runs as smoothly as it should. In reality these technologies do not always work perfectly together, potentially creating confusion for both the user and the analyst in the interaction process. The key to understanding system behaviour is once more to be found in the system logs. An example from the case study of treasure (section 5.2, which was deliberately created to play with the ‘edges’ or ‘seams’ of interacting technologies [47] is to be found in section 5.2.1 where the GPS system was reporting one position to a user, but disconnection in the messenger system meant that the server recorded them at an entirely different position (something invisible to the users) caused a breakdown in the player’s understanding of the game. One might assume that the server has a complete picture of the ‘state’ of the game, but in practice this is not the case. Position reports can only happen when the PDA is within network range of the server, but the game requires players to leave the network, so its ‘state’ can only ever be a ‘best guess’ based on last known location. However, because the PDA also records its position, when we combine the logs into an aggregated spatial distribution we can see the discrepancies in these locations. An interview with the user suggested they were attempting one tactic, while the system logs suggested otherwise. Each viewpoint (user, PDA, server) tells a different story, and the facts can only be reconciled by combining those viewpoints. This provides a specific example of how the complete picture can only be found in the coordinated examination of both the system log data and the more traditionally recorded interview data - either alone would not have served to explain this discontinuity.

### 6.2.9 Meeting the Challenges

We have seen in the examples above how the practice of coordinating system logs with more traditional data can alleviate many of the problems associated with studying ubiquitous computing systems. It is notable that the programs do not in themselves solve the problems, it is by understanding their potential uses through the whole process of designing an

experiment, deciding what to record and log, running the experiment then collecting, synchronising and analysing the data, that an analyst can exploit these facilities to overcome the challenges laid out in the related work chapter. Computer aided qualitative data analysis tools such as Replayer and DRS are not in themselves a solution to these challenges. Indeed such tools cannot and should not replace the core skills of a qualitative data researcher, rather they provide a way of exploring and exploiting new types of data to which traditional qualitative methods may be applied. Just as word processors provide many valuable resources to an author, the skill, process, methods and conclusions remain the province of the individual researchers. Replayer and DRS simply help the researcher to apply his or her methods across a wider range of data to reflect the wider range of interaction media the researcher is faced with the challenge of understanding.

### 6.3 Revisiting the design guidelines

At the end of the related work section we laid out a series of key design guidelines necessary for any system aiming to support qualitative data analysis of ubiquitous computing systems, in particular through the use of system log data as a potential resource for qualitative description. We shall now show how Replayer and DRS have been developed within those guidelines.

- *Tools that allow viewing of system log data synchronized with other types of media.*

We have examined in detail in chapters 3 and 4 how both Replayer and DRS provide support for the viewing of system log data synchronized with additional source media such as video and audio in the case of Replayer and with additional media such as transcriptions, photographs and documents in DRS. We have seen in the case studies how the use of those logs in coordination with other media types can inform an analysis and support the description of interaction with and around ubiquitous computing technology.

- *Tools that enable researchers to extract multiple sources of information from recorded logs, and which allow them to edit extracted information and combine it with media from external sources to produce unique datasets.*

The case studies described in chapter 5 and in section 6.2 have examined the use of different types of logged data and how that data can be used in coordination with other

data to create uniquely powerful resources for qualitative description. We have further shown how it is possible to extract certain information *only* by using the combination of both media types.

- *A replay system that exploits time stamps to coordinate the use of the multiple media in a dataset, which enables cross referencing and indexing to support the splicing together of multiple media, and which enables multiple media to played side-by-side.* Both Replayer and DRS provide the ability to synchronize multiple media sources on a single timeline. DRS in particular demonstrates by use of coding (Track Viewer, Gesture Recognition) and document markup (DRS Documents) how heterogeneous media types can be effectively cross referenced. Both systems provide means by which one media type, or view of a media type can be used as an index to another.
- *Tools that support the production of representations from datasets and which preserve the relationship between representations and the media from which they are derived, and which enable source media to be recovered and viewed.*

In the case of Replayer we see many potential representations of system log data with the set of graphical viewers provided. In DRS we can see how more familiar approaches such as transcription and coding can also be used to create synchronized representations of the data both at an absolute and descriptive level. In all these cases the coordinated nature of the data handling allows source media to be instantly accessible even after abstraction, such as in the case of Replayer’s correlation tool.

Both Replayer and DRS have been implemented with these requirements at the core of their design.

## 6.4 Specific Academic Contributions

This section will describe the specific academic contributions made by this thesis and the work described therein.

- *Demonstrated the value that can be added by combining log data with other media.* We have shown through case studies that using system log data as a qualitative resource for can add significant value to an analysis. While the contents of the log file do not by themselves represent qualitative data, the application of those resources to can help to determine the character of an interaction that may be otherwise difficult both to

study and to describe because of the inherent difficulties in working with ubiquitous computing tools as laid down in the challenges. Specifically we have shown that the coordination of those *accountable* logs with other data types such as video and audio can, when a suitable qualitative research methodology such as ethnomethodologically informed ethnography is applied, serve as an effective descriptive resource.

- *Shown the coordination of system logs and other media can help us ‘see things we could not otherwise have seen.’* Closely related to the above point, we have demonstrated in the case studies examples of the type of value added by system log data, in that in the coordination of that data with other media it becomes possible to understand interaction in ways that would have been impossible (or at least extremely difficult) in any other way. As specific example of this can be seen in the Treasure case study during the discussion on inaccurate position displays. The player’s understanding of the system and the technical implementation of the system did not quite match. Because the player was unfamiliar with the technical details of the system, all they see is a problem - which is highlighted in interviews with the player by the evaluator. Only by interacting with the logs, and using multiple views over those logs can this problem be effectively described. In particular this example serves to highlight several of the key challenges discussed above: the players are mobile, the device they use has a small display, they are interacting with other players who are not collocated, and they are interacting with an imperfect network (a seamful environment). All of these issues are overcome by the coordinated use of system logs and other media in a synchronized playback environment, allowing effective description of the players’ actual interaction with the system and with each other.
- *Ability to perform studies in locations/settings that would otherwise have been impossible* The very nature of ubiquitous computing systems can make the settings in which their use occurs difficult to study with traditional qualitative methods. The use of recorded log data along with other heterogeneous media can allow researchers to apply those qualitative methods by adding previously invisible data as a coordinated description resource. For example in the Day of the Figurines case study (section 5.3), we are unable to directly observe all the diversely located players over a long period of time (four weeks), however we are able to reconstruct the events that occurred in the game by making a representation of the system logs. We can then interrogate

this representation like any other more traditional type of observation and unpack the recorded events.

- *Demonstrated how computer aided qualitative data analysis software has to evolve to support understanding interaction in a world increasing populated with ubiquitous computing technology* Having explored the current state of the art in CAQDAS tools, we have then demonstrated by means of the difficulties laid down in the related work section, how the current tools are simply not equipped to cope with the new challenges of studying the use of and the interaction around ubiquitous computing technology. We have demonstrated through the development of both Replayer and Digital Replay System, how those tools can be evolved to incorporate additional system log information as well as ways by which that data can be viewed, explored and exploited as an integrated resource for qualitative analysis.
- *Demonstrated how creating accessible qualitative data analysis tools allows people to study particular settings or technologies that could not have studied them before.* We have demonstrated that by creating familiar tools, ubiquitous computing can be studied by ‘real’ social scientists, and not just computing scientists. The complexity of system log data can be reduced to manageable and accountable data by means of coordinated visualization. In effect this means that analysis of ubiquitous computing systems can be effectively performed by those who are experts in qualitative analysis, instead of those whose expertise lies in the technical aspects of ubiquitous computing.
- *Defined, implemented and demonstrated an effective framework for storing and synchronizing log files.* The organizational paradigm of state and event logs as demonstrated in the implementation of Replayer serves as a powerful framework for storing and handling high dimensional system log data. The framework is sufficiently flexible to handle every type of system log yet encountered in the development process, and many others besides. This framework allows sampled data to interact with recorded events so if suitable data are available, a system’s state can be framed at any given time that an event occurs. The framework is sufficiently generalized to be able to include not just system log data, but any annotations and by means of that also coding and transcription data. This means that recorded data, whether it be system logs, or descriptive data added post-hoc, can be coordinated, searched and explored together.
- *Shown how this framework can be exploited to achieve high dimensional synchroniza-*



*tion of recorded data.* We have demonstrated how with the use of a series of coordinated views, it is possible to make complex selections within a given dataset achieving synchronization over several dimensions beyond just time. Within the framework of DRS we have shown that the use of ‘markup’ of documents to include temporal information, and links to additional media allows us to ‘synchronize’ a document that in itself has no inherent temporal structure.

- *Shown that with the use of a distributed software architecture some of the basic problems associated with viewing high dimensional data can be addressed* Replayer’s distributed architecture serves to provide a flexible platform for playing back data across multiple computer systems, helping to circumvent problems such as screen real estate or limited processing power. as well as supporting language independent extensibility. DRS’s distributed architecture further supports the task of corpus management, allowing individuals to interact with different corpora with appropriate security measures.
- *Co-developed two proof of concept pieces of software, Replayer and Digital Replay System with radically different approaches to supporting the use of heterogeneous media types as a resource for qualitative analysis of interaction with and around ubiquitous computing systems, and shown how those disparate approaches can be combined to create the next generation of computer aided qualitative data analysis software.*

## 6.5 Conclusion

The study of ubiquitous computing systems presents significant challenges that have not previously been encountered in interaction analysis of desktop systems and virtual environments. Ubiquitous computing situates users in a heterogeneous array of physical and digital environments; users interact via different interaction mechanisms; and interaction itself is mediated by invisible sensing systems. The *asymmetrical, fragmented, and invisible* nature of ubiquitous computing [32], makes interaction difficult to observe from the outset. The challenge is often treated as a methodological one: a matter of developing analytic frameworks and methods that are capable of handling the problem of understanding interaction that ubiquitous computing brings with it (e.g. [86,93,99]). Efforts to address the problem adapt existing techniques, quantitative, qualitative, experimental and naturalistic. The ubiquitous computing and HCI literature is replete with examples that together form a consensus that the problem of understanding interaction in ubiquitous computing environments is a

methodological problem through and through. There is more to the matter than method, however. Before we can devise and employ a methodical way of observing interaction, we need to be able to see it. Yet interaction in ubiquitous computing environments is mobile, massively distributed, located on small devices, mediated through invisible sensing systems etc. In other words, the problem is not so much that we do not have suitable methods but that we cannot see what is going on, at least to a sufficient degree in the first instance. The problem is not a methodological one per se, but an observational one or, to put it another way, before we address issues of method, we first need to address the visibility of the phenomenon.

The need to make the phenomenon visible is an old and constant preoccupation of scientific endeavour [44]. Our understanding of phenomena great and small depends on the development of technologies that make the phenomenon visible: microscopes, telescopes, particle accelerators, etc. The sciences are replete with technologies of observation. The advent of ubiquitous computing places the same demand on computer science if we are to develop an adequate understanding of human interaction within increasingly complex technological environments: as the computer disappears, our interactions become increasingly opaque.

It has been the work of this thesis to demonstrate through in depth literature review; the development of core requirements for new observational tools; the subsequent development of two example software packages (*Replayer* and *Digital Replay System*); two in depth case studies of their use in practice and reflection on what value they have shown to add to an analysis, that such technologies of observation for ubiquitous computing systems are not only feasible but practicable and necessary. A large part of this work has been the exploitation of system logs - in particular creating accountable representations of those logs which can become a significant qualitative resource.

For all the added extras that they bring however, system logs do not, in and of themselves, contain data. Rather, the data must be produced through the analytically oriented working of resources internal to situated action and their combination with resources external to the setting of action. Data is constructed then and produced through the work practices of the analyst. The identification and extraction of salient features enables the analyst to start to make sense of the system logs, and through the understanding, filtering and visualization of those logs to construct and characterize interaction in and around ubiquitous computing systems.

Just as the process of capturing data changed as photographs and video became an ac-

cessible resource, so too it must change to accommodate this ‘born digital’ data. Ubiquitous computing is only going to increase in volume and pervasion over time and it is important that social scientists adopt the necessary resources and processes to understand it and exploit both the technology itself, and the data captured from such technology as one more facet in an ever expanding collection of resource types.

# Bibliography

- [1] CIA: The World Factbook. <https://www.cia.gov/cia/publications/factbook/index.html>, 2007.
- [2] G D Abowd. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, 38(4):508–530, 1999.
- [3] G D Abowd, C G Atkeson, A F Bobick, I A Essa, B MacIntyre, E D Mynatt, and T E Starner. Living laboratories: the future computing environments group at the Georgia Institute of Technology. In *CHI'00 extended abstracts on Human factors in computing systems*, pages 215–216. ACM, 2000.
- [4] G D Abowd, L D Harvel, and J A Brotherton. Building a digital library of captured educational experiences. In *Digital Libraries: Research and Practice, 2000 Kyoto, International Conference on.*, pages 467–474. IEEE, 2000.
- [5] G D Abowd, E D Mynatt, and T Rodden. The human experience [of ubiquitous computing]. *Pervasive Computing, IEEE*, 1(1):48–57, 2002.
- [6] S Adolphs and R Carter. Beyond the word: new challenges in analysing corpora of spoken English. *European Journal of English Studies*, 11(2):114–128, 2007.
- [7] Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th international conference on World Wide Web*, pages 529–535. ACM, 2003.
- [8] Ghassan Al-Qaimari and Darren McRostie. KALDI: A Computer-Aided Usability Engineering Tool for Supporting Testing and Analysis of Human-Computer Interaction. In *CADUI*, pages 337–355, 1999.

- [9] Ian Anderson, Julie Maitland, Scott Sherwood, Louise Barkhuus, Matthew Chalmers, Malcolm Hall, Barry Brown, and Henk Muller. Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mob. Netw. Appl.*, 12(2-3):185–199, 2007.
- [10] Chee Siang Ang, Ania Bobrowicz, Panote Siriaraya, Joshua Trickey, and Kate Winspear. Effects of gesture-based avatar-mediated communication on brainstorming and negotiation tasks among younger users. *Computers in Human Behavior*, null(null), November 2012.
- [11] Albert N Badre, Mark Guzdial, Scott E Hudson, and Paulo J Santos. A user interface evaluation environment using synchronized video, visualizations and event trace data. *Software Quality Journal*, 4(2):101–113, 1995.
- [12] Liam J Bannon. *From Human Factors to Human Actors: The Role of Psychology and Human-Computer Interaction Studies in System Design*, chapter 2, pages 25–45. Lawrence Erlbaum Associates, Inc, New Jersey, USA, 1991.
- [13] J E Bardram. Hospitals of the future—ubiquitous computing support for medical work in hospitals. In *UbiHealth*, 2003.
- [14] Louise Barkhuus, Matthew Chalmers, Paul Tennent, Malcolm Hall, Marek Bell, Scott Sherwood, and Barry Brown. Picking Pockets on the Lawn: The Development of Tactics and Strategies in a Mobile Game. In Michael Beigl, Stephen S Intille, Jun Rekimoto, and Hideyuki Tokuda, editors, *Ubicomp*, volume 3660 of *Lecture Notes in Computer Science*, pages 358–374. Springer, 2005.
- [15] L Bartram, C Ware, and T Calvert. Moving icons: Detection and distraction. In *Proc. IFIP TC*, volume 13, pages 157–165, 2001.
- [16] Duncan Bates, Nigel Linge, David Parsons, Robin Holgate, Pauline Webb, David Hay, Sian Wynn-Jones, Alex Newson, and David Ward. Building context into a museum information guide. pages 235–241, October 2007.
- [17] J B Bavelas and N Chovil. Nonverbal and Verbal Communication: Hand Gestures and Facial Displays as Part of Language Use in Face-to-face Dialogue. 2006.

- [18] Geoffrey Beattie and Heather Shovelton. What properties of talk are associated with the generation of spontaneous iconic hand gestures? *British Journal of Social Psychology*, 41(3):403–417, September 2002.
- [19] Richard A Becker and William S Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.
- [20] M Bell, M Chalmers, B Brown, I MacColl, M Hall, and P Rudman. Sharing photos and recommendations in the city streets. In *Pervasive 2005 Workshop on Exploring Context Histories in Smart Environments (ECHISE)*. N/A, 2005.
- [21] Marek Bell, Matthew Chalmers, Louise Barkhuus, Malcolm Hall, Scott Sherwood, Paul Tennent, Barry Brown, Duncan Rowland, and Steve Benford. Interweaving mobile games with everyday life. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 417–426, New York, NY, USA, 2006. ACM.
- [22] V Bellotti and I Smith. Informing the design of an information management system with iterative fieldwork. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 227–237. ACM, 2000.
- [23] S Benford, W Seagar, M Flintham, R Anastasi, D Rowland, J Humble, D Stanton, J Bowers, N Tandavanitj, M Adams, Row J Farr, A Oldroyd, and J Sutton. The Error of our Ways: The experience of Self-Reported Position in a Location-Based Game. In *Proceedings of the the 6th International Conference on Ubiquitous Computing. (UbiComp 2004)*, pages 70–87, Nottingham, September 2004.
- [24] Steve Benford, Rob Anastasi, Martin Flintham, Adam Drozd, Andy Crabtree, Chris Greenhalgh, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Coping with Uncertainty in a Location-Based Game. *IEEE Pervasive Computing*, 02(3):34–41, 2003.
- [25] Steve Benford, Andy Crabtree, Martin Flintham, Adam Drozd, Rob Anastasi, Mark Paxton, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Can you see me now? *ACM Trans. Comput.-Hum. Interact.*, 13(1):100–133, 2006.
- [26] Adriana Holtz Betiol and Walter de Abreu Cybis. Usability Testing of Mobile Devices: A Comparison of Three Approaches. *Human-Computer Interaction - INTERACT 2005*, pages 470–481, 2005.

- [27] E Bittner. Objectivity and realism in sociology. In G Psathas, editor, *Phenomenological Sociology*, pages 109–125. John Wiley, 1973.
- [28] S Björk, J Falk, R Hansson, and I. Ljungstrand. Pirates! - Using the Physical World as a Game Board. In *IFIP TC.13 Conference on Human-Computer Interaction (INTERACT 2001)*, 2001.
- [29] Jesse Blum, Martin Flintham, Rachel Jacobs, Victoria Shipp, Genovefa Kefalidou, Michael Brown, and Derek McAuley. The timestreams platform: Artist mediated participatory sensing for environmental discourse. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 285–294, New York, NY, USA, 2013. ACM.
- [30] B Brown and I MacColl. Lessons from the lighthouse: Collaboration in a shared mixed reality system, 2003.
- [31] Barry Brown and Matthew Chalmers. Tourism and mobile technology. In *ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, pages 335–354, Norwell, MA, USA, 2003. Kluwer Academic Publishers.
- [32] Barry Brown, Matthew Chalmers, Marek Bell, Malcolm Hall, Ian MacColl, and Paul Rudman. Sharing the square: collaborative leisure in the city streets. In *ECSCW'05: Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, pages 427–447, New York, NY, USA, 2005. Springer-Verlag New York, Inc.
- [33] H Brugman and A Russel. Annotating multi-media / multi-modal resources with ELAN. In *LREC2004*, pages 2065–2068, 2004.
- [34] Patrick Brundell, Dawn Knight, Paul Tennent, A Naeem, Svenja Adolphs, Shaaron Ainsworth, Ronald Carter, David Clarke, Andrew Crabtree, Chris Greenhalgh, Claire O'Malley, T Pridmore, and Tom Rodden. The experience of using the Digital Replay System for social science research. In *4th International e-Social Science Conference*, Manchester, UK, June 2008.
- [35] Patrick Brundell, Paul Tennent, Chris Greenhalgh, Dawn Knight, Andrew Crabtree, Claire O'Malley, Shaaron Ainsworth, David Clarke, Ronald Carter, and Svenja

- Adolphs. Digital Replay System (DRS): A Tool for Interaction Analysis. In *International Conference on Learning Sciences (Workshop on Interaction Analysis)*, Utrecht, June 2008.
- [36] Brandon Burr. VACA: a tool for qualitative video analysis. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 622–627, New York, NY, USA, 2006. ACM.
- [37] G Burrell, J and Gay. E-Graffiti. *Interacting with Computers*, 14:301–312, 2002.
- [38] Graham Button and Paul Dourish. Technomethodology. In *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*, pages 19–26, New York, New York, USA, April 1996. ACM Press.
- [39] John T. Cacioppo, Louis G. Tassinary, and Gary Berntson, editors. *Handbook of Psychophysiology*. Cambridge University Press, 2007.
- [40] Mauricio Capra, Milena Radenkovic, Steve Benford, Leif Oppermann, Adam Drozd, and Martin Flintham. The multimedia challenges raised by pervasive games. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 89–95, New York, NY, USA, 2005. ACM.
- [41] J Carletta, S Evert, U Heid, J Kilgour, J Robertson, and H Voormann. The NITE XML Toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers*, 35(3), 2003.
- [42] R Carter and S Adolphs. Linking the verbal and the visual: new directions for corpus linguistics. *Language and Computers*, 64:275–291, 2008.
- [43] Scott Carter and Jennifer Mankoff. Prototypes in the Wild: Lessons from Three Ubicomp Systems. *IEEE Pervasive Computing*, 4(4):51–57, 2005.
- [44] Scott Carter, Jennifer Mankoff, and Jeffrey Heer. Memento: support for situated ubicomp experimentation. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 125–134, New York, NY, USA, 2007. ACM.
- [45] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world’s largest user generated content



- video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.
- [46] Matthew Chalmers, Marek Bell, Malcolm Hall, Scott Sherwood, and Paul Tennent. Seamful Games. *Adjunct Proceedings of 6th Int. Conf. on Ubiquitous Computing*, pages 7–10, 2004.
- [47] Matthew Chalmers and Areti Galani. Seamful interweaving: heterogeneity in the theory and design of interactive systems. In *DIS '04: Proceedings of the 5th conference on Designing interactive systems*, pages 243–252, New York, NY, USA, 2004. ACM.
- [48] Purnima Chawla and Robert M. Krauss. Gesture and Speech in Spontaneous and Rehearsed Narratives. *Journal of Experimental Social Psychology*, 30(6):580–601, November 1994.
- [49] C Chewar and D S McCrickard. Adapting UEMS for notification systems. In *UbiComp 2002 Workshop*, volume 9, 2002.
- [50] Michael G. Christel, Scott M. Stevens, Bryan S. Maher, and Julieanna Richardson. Enhanced exploration of oral history archives through processed video and synchronized text transcripts. In *Proceedings of the International Conference on Multimedia, MM '10*, pages 1333–1342, New York, NY, USA, 2010. ACM.
- [51] S Consolvo, L Arnstein, and B Franza. User study techniques in the design and evaluation of a ubicomp environment. *UbiComp 2002: Ubiquitous Computing*, pages 281–290, 2002.
- [52] S. Consolvo and M. Walker. Using the experience sampling method to evaluate ubicomp applications. *IEEE Pervasive Computing*, 2(2):24–31, April 2003.
- [53] Louise Corti. Qualitative Data Exchange: Methods and Tools. In *Association for Survey Computing: The Challenges of a Changing World (ASC2007)*, 2007.
- [54] A Crabtree, T Rodden, T Hemmings, and S Benford. Finding a Place for UbiComp in the Home. In *UbiComp 2003: Ubiquitous Computing*, pages 208–226. Springer, 2003.
- [55] A Crabtree and M Rouncefield. Working with text logs: some early experiences of record and replay. In *1st International e-Social Science Conference*, Manchester, UK, 2005.

- [56] Andy Crabtree. *Designing Collaborative Systems: A Practical Guide to Ethnography (Google eBook)*. Springer, 2003.
- [57] Andy Crabtree, Steve Benford, Mauricio Capra, Martin Flinham, Adam Drozd, Nick Tandavanitj, Matt Adams, and Ju Row Farr. The Cooperative Work of Gaming: Orchestrating a Mobile SMS Game. *Comput. Supported Coop. Work*, 16(1-2):167–198, 2007.
- [58] Andy Crabtree, Steve Benford, Chris Greenhalgh, Paul Tennent, Matthew Chalmers, and Barry Brown. Supporting ethnographic studies of ubiquitous computing in the wild. In *Proceedings of the 6th ACM conference on Designing Interactive systems - DIS '06*, page 60, New York, New York, USA, June 2006. ACM Press.
- [59] H. D. Critchley. Book Review: Electrodermal Responses: What Happens in the Brain. *The Neuroscientist*, 8(2):132–142, April 2002.
- [60] Edward Cutrell, Mary Czerwinski, and Eric Horvitz. Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance. In *Human-computer Interaction: INTERACT'01: IFIP TC. 13 International Conference on Human-Computer Interaction, 9th-13th July 2001, Tokyo, Japan*, pages 263–269. IOS Press, 2001.
- [61] David Cwir, Priyanka B. Carr, Gregory M. Walton, and Steven J. Spencer. Your Heart Makes My Heart Move: Cues of Social Connectedness Cause Shared Emotions and Physiological States Among Strangers. *Journal of Experimental Social Psychology*, 47(3):664–661, January 2011.
- [62] M Czyzewski. Reflexivity of actors versus the reflexivity of accounts. *Theory, Culture and Society*, 11:161–168, 1994.
- [63] N Dahlbäck, A Jönsson, and L Ahrenberg. Wizard of Oz studies: why and how. *Knowledge-based systems*, 6(4):258–266, 1993.
- [64] Norman K. Denzin and Yvonna S. Lincoln. Introduction: The discipline and practice of qualitative research. In Norman K. Denzin and Yvonna S. Lincoln, editors, *The Sage Handbook of Qualitative Research*, pages 1–33. Sage, Thousand Oaks, CA, 3 edition, 2005.

- [65] Silvana di Gregorio and Judith Davidson. Research Design, Units of Analysis and Software supporting Qualitative Analysis. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.
- [66] Tanja Döring, Alireza Sahami Shirazi, and Albrecht Schmidt. Exploring gesture-based interaction techniques in multi-display environments with mobile phones and a multi-touch table. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI '10*, pages 419–419, New York, NY, USA, 2010. ACM.
- [67] Paul Dourish. *Where the Action Is : The Foundations of Embodied Interaction (Bradford Books)*. The MIT Press, September 2004.
- [68] S Dumais and M Czerwinski. Building bridges from theory to practice. In *HCI International 2001, 9th Conference on Human-Computer Interaction*, 2001.
- [69] G Dyke, J Girardot, K Lund, and A Corbel. Analysing face to face computer-mediated interactions. In *EARLI (European Association for Research, Learning and Instruction), 12th Biennial International Conference*, 2007.
- [70] P Ekman and W V Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *Nonverbal communication, interaction, and gesture*, pages 57–106, 1981.
- [71] Paul Ekman and Wallace V Friesen. Nonverbal leakage and clues to deception. Technical report, DTIC Document, 1969.
- [72] Paul Ekman and Maureen OSullivan. Facial expression: Methods, means, and moues. *Fundamentals of nonverbal behavior*, 1:163–199, 1991.
- [73] Paul Ekman and Erika L Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 1997.
- [74] Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim Mccall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 599–607, New York, NY, USA, 1992. ACM Press.

- [75] D Evans and A Naeem. Using visual tracking to link text and gesture in studies of natural discourse. In *Online Proceedings of the Cross Disciplinary Research Group Conference Exploring Avenues to Cross-Disciplinary Research, November*, volume 7, 2007.
- [76] Jerry Alan Fails, Allison Druin, and Mona Leigh Guha. Mobile collaboration. In *Proceedings of the 9th International Conference on Interaction Design and Children - IDC '10*, page 20, New York, New York, USA, June 2010. ACM Press.
- [77] Xiaoli Fern, Chaitanya Komireddy, Valentina Grigoreanu, and Margaret Burnett. Mining problem-solving strategies from hci data. *ACM Trans. Comput.-Hum. Interact.*, 17(1):3:1–3:22, April 2010.
- [78] Martin Flintham, Rob Anastasi, Steve Benford, Adam Drozd, James Mathrick, Duncan Rowland, Amanda Oldroyd, Jon Sutton, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Uncle Roy all around you: mixing games and theatre on the city streets. In *DIGRA Conf.*, 2003.
- [79] Ian Foster and Carl Kesselman. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, November 1998.
- [80] Adam Fouse, Nadir Weibel, Edwin Hutchins, and James D. Hollan. ChronoViz: a system for supporting navigation of time-coded data. pages 299–299–304–304, May 2011.
- [81] Adam S Fouse and James D Hollan. Visualization of exploratory video analysis.
- [82] Mike Fraser, Jon Hindmarsh, Katie Best, Christian Heath, Greg Biegel, Chris Greenhalgh, and Stuart Reeves. Remote Collaboration Over Video Data: Towards Real-Time e-Social Science. *Comput. Supported Coop. Work*, 15(4):257–279, 2006.
- [83] Mike Fraser, Jon Hindmarsh, Katie Best, Christian Heath, Greg Biegel, Chris Greenhalgh, and Stuart Reeves. Remote Collaboration Over Video Data: Towards Real-Time e-Social Science. *Computer Supported Cooperative Work (CSCW)*, 15(4):257–279, September 2006.
- [84] Christopher Frauenberger, Judith Good, Alyssa Alcorn, and Helen Pain. Supporting the design contributions of children with autism spectrum conditions. In *Proceedings*

- of the 11th International Conference on Interaction Design and Children - IDC '12, page 134, New York, New York, USA, June 2012. ACM Press.
- [85] A French, C Greenhalgh, A Crabtree, M Wright, P Brundell, A Hampshire, and T Rodden. Software replay tools for time-based social science data. In *(ICESS2006) 2nd International Conference on e-Social Science*, 2006.
  - [86] Jon Froehlich, Mike Y Chen, Sunny Consolvo, Beverly Harrison, and James A Landay. MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 57–70, New York, NY, USA, 2007. ACM.
  - [87] Thomas M J Fruchterman and Edward M Reingold. Graph Drawing by Force-directed Placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.
  - [88] H Garfinkel. *Studies in Ethnomethodology*. Prentice-Hall, Englewood Cliffs, NJ, 1967.
  - [89] H Garfinkel. Hybrid Studies. In A Rawls, editor, *Ethnomethodology's Program: Working Out Durkheim's Aphorism*, pages 100–103. Rowman and Littlefield, 2001.
  - [90] Harold Garfinkel and Harvey Sacks. On Formal Structures of Practical Action. In J McKinney and E Tiryakian, editors, *Theoretical Sociology*, pages 337–366. New York: Appleton-Century-Crofts, 1970.
  - [91] R Stuart Geiger and David Ribes. The work of sustaining order in wikipedia: the banning of a vandal. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 117–126. ACM, 2010.
  - [92] M Gellner and F Forbrig. No Title. In *HCI International, Adjunct Workshop*, 2003.
  - [93] M Gellner and P Forbrig. ObSys—A tool for visualizing usability evaluation patterns with mousemaps. In *Proc. of the Tenth Int. Conf. on Human-Computer Interaction*, pages 469–473, 2003.
  - [94] Olivier Georgeon, Matthias J Henning, Thierry Bellet, and Alain Mille. Creating Cognitive Models from Activity Analysis: A Knowledge Engineering Approach to Car Driver Modeling. In *International Conference on Cognitive Modeling*, pages 43–48. Taylor & Francis, July 2007.

- [95] Gabriella Giannachi, Henry Lowood, Glen Worthey, Dominic Price, Duncan Rowland, and Steve Benford. Documenting mixed reality performance: the case of cloudpad. *Digital Creativity*, 23(3-4):159–175, 2012.
- [96] Gabriella Giannachi, Duncan Rowland, Steve Benford, Jonathan Foster, Matt Adams, and Alan Chamberlain. Blast Theory’s Rider Spoke , its Documentation and the Making of its Replay Archive. *Contemporary Theatre Review*, 20(3):353–367, August 2010.
- [97] Jean Ann Graham and Michael Argyle. A Cross-Cultural Study of the Communication of Extra-Verbal Meaning by Gesture. *International Journal of Psychology*, 10(1):57–67, January 1975.
- [98] C Greenhalgh. EQUIP: a Software Platform for Distributed Interactive Systems, 2002.
- [99] Ian Hacking. *Representing and Intervening*. Cambridge University Press, Cambridge, 1983.
- [100] William A. Hamilton, Zachary O. Toups, and Andruid Kerne. Synchronized communication and coordinated views: Qualitative data discovery for team game user studies. In *CHI ’09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’09, pages 4573–4578, New York, NY, USA, 2009. ACM.
- [101] Eric Harris, Geraldine Fitzpatrick, Yvonne Rogers, Sara Price, Ted Phelps, and Cliff Randell. From snark to park: lessons learnt moving pervasive experiences from indoors to outdoors. In *AUIC ’04: Proceedings of the fifth conference on Australasian user interface*, pages 39–48, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
- [102] C Heath and J Hindmarsh. Analysing Interaction. *Video Ethnography*, 2002.
- [103] C Heath and J Hindmarsh. *Analyzing interaction: video, ethnography and situated conduct*, volume 99-121. Sage, 2002.
- [104] Martin Heidegger. *Being and Time (Translation)*. Blackwell Publishers, May 1996.
- [105] Sharlene Hesse-Biber and Christine Crofts. Computer-Aided Software for Qualitative Data Analysis: An Historical Overview and Contemporary Perspectives. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.

- [106] David M Hilbert and David F Redmiles. Extracting usability information from user interface events. *ACM Computing Surveys*, 32(4):384–421, 2000.
- [107] James D. Hollan. Activity-enriched computing: Capturing and mining activity histories. *Computer*, 45(10):84–87, October 2012.
- [108] J G Hollands and C D Wickens. *Engineering psychology and human performance*. Prentice Hall New Jersey, 1999.
- [109] L E Holmquist, K Höök, O Juhlin, and P Persson. Challenges and opportunities for the design and evaluation of mobile applications. In *workshop Main issues in designing interactive mobile services, Mobile HCI*, volume 2002, 2002.
- [110] Michael Horn, Zeina Atrash Leong, Florian Block, Judy Diamond, E. Margaret Evans, Brenda Phillips, and Chia Shen. Of BATs and APes. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 2059, New York, New York, USA, May 2012. ACM Press.
- [111] Juan Pablo Hourcade, Natasha E. Bullock-Rest, and Thomas E. Hansen. Multitouch tablet applications and activities to enhance the social skills of children with autism spectrum disorders. *Personal and Ubiquitous Computing*, 16(2):157–168, April 2011.
- [112] Philip N Howard. Network ethnography and the hypermedia organization: New media, new organizations, new methods. *New Media & Society*, 4(4):550–574, 2002.
- [113] S Hudson, J Fogarty, C Atkeson, D Avrahami, J Forlizzi, S Kiesler, J Lee, and J Yang. Predicting human interruptibility with sensors: a Wizard of Oz feasibility study. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 257–264. ACM, 2003.
- [114] Robert L Hulsman, Ellen M A Smets, John M Karemaker, and Hanneke J C J M de Haes. The psychophysiology of medical communication. Linking two worlds of research. *Patient education and counseling*, 84(3):420–7, September 2011.
- [115] Giovanni Iachello, Khai N Truong, Gregory D Abowd, Gillian R Hayes, and Molly Stevens. Prototyping and sampling experience to evaluate ubiquitous computing privacy in the real world. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1009–1018, New York, NY, USA, 2006. ACM.

- [116] S Intille. Change blind information display for ubiquitous computing environments. *UbiComp 2002: Ubiquitous Computing*, pages 193–222, 2002.
- [117] Stephen S Intille, John Rondoni, Charles Kukla, Isabel Ancona, and Ling Bao. A context-aware experience sampling tool. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 972–973, New York, NY, USA, 2003. ACM.
- [118] Melody Y Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.
- [119] Rachel Jacobs, Steve Benford, Mark Selby, Michael Golembewski, Dominic Price, and Gabriella Giannachi. A conversation between trees: What data feels like in the forest. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13*, pages 129–138, New York, NY, USA, 2013. ACM.
- [120] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007.
- [121] X Jiang, J I Hong, L A Takayama, and J A Landay. Ubiquitous computing for firefighters: Field studies and prototypes of large displays for incident command. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 679–686. ACM, 2004.
- [122] Yasmin Kafai and Deborah Fields. Connecting play: Understanding multimodal participation in virtual worlds. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction, ICMI '12*, pages 265–272, New York, NY, USA, 2012. ACM.
- [123] A Kaikkonen, T Kallio, A Keklinen, A Kankainen, and M Cankar. Usability testing of mobile applications: A comparison between laboratory and field testing. *Journal of Usability Studies*, 1:4–16.
- [124] Dimitris N. Kanellopoulos. Current and future directions of multimedia technology in tourism. *International Journal of Virtual Technology and Multimedia*, 1(2):187, March 2010.



- [125] A Kendon. Some Emerging Features of Face-to-Face Interaction Studies. *Sign Language Studies*, 22:7–22, 1979.
- [126] A Kendon. *Conducting Interaction: Patterns of behavior in focused encounters*, volume 7. Cambridge University Press, 1990.
- [127] A Kendon. Do gestures communicate? A review. *Research on language and social interaction*, 27(3):175–200, 1994.
- [128] A Kendon, R M Harris, and M R Key. *Organization of behavior in face-to-face interaction*. De Gruyter Mouton, 1975.
- [129] D Kieras and S Bovair. The role of a mental model in learning to operate a device. *Cognitive Science*, 8(3):255–273, September 1984.
- [130] Michael Kipp. Anvil - A Generic Annotation Tool for Multimodal Dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, Aalborg, September 2001.
- [131] Jesper Kjeldskov, Mikael B Skov, Benedikte S Als, and Rune T Huegh. Is It Worth the Hassle? Exploring the Added Value of Evaluating the Usability of Context-Aware Mobile Systems in the Field. *Mobile Human-Computer Interaction (MobileHCI 2004)*, pages 61–73, 2004.
- [132] D Knight. Corpora: the next generation. *Part of the AHRC funded online Introduction to Corpus Investigative Techniques, The University of Birmingham*. <http://www.humcorp.bham.ac.uk>, 2006.
- [133] Dawn Knight and Paul Tennent. Introducing Drs: A Tool For The Future Of Corpus Linguistic Research And Analysis. In *The 6th Language Resources And Evaluation Conference*. Elra, 2008.
- [134] Jan-Peter Krämer, Thorsten Karrer, Joachim Kurz, Moritz Wittenhagen, and Jan Borchers. How tools in ides shape developers’ navigation behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’13*, pages 3073–3082, New York, NY, USA, 2013. ACM.
- [135] A Kranstedt, S Kopp, and I Wachsmuth. MURML: A multimodal utterance representation markup language for conversational agents. In *Proc. of the AAMAS Workshop on Embodied conversational agents—Lets specify and evaluate them*, 2002.

- [136] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.
- [137] Steinar Kristoffersen and Fredrik Ljungberg. "Making place" to make IT work: empirical explorations of HCI for mobile CSCW. In *GROUP '99: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 276–285, New York, NY, USA, 1999. ACM.
- [138] Udo Kuckartz. Techniques of analysis using MAXQDA 07. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.
- [139] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: device positioning using radio beacons in the wild. In *Proceedings of the Third international conference on Pervasive Computing*, PERVASIVE'05, pages 116–133, Berlin, Heidelberg, 2005. Springer-Verlag.
- [140] Shannon Lane and Emmy Arnold. Qualitative research: a valuable tool for transfusion medicine. *Transfusion*, 51(6):1150–3, June 2011.
- [141] R Larson and M Csikszentmihalyi. The Experience Sampling Method. In H T Reis, editor, *Naturalistic Approaches to Studying Social Interaction: New Directions for Methodology of Social and Behavioral Science*. Jossey-Bass, 1983.
- [142] Dominique Le-Roux, Magda Dargentas, and Mathieu Brugidou. Developing Computer-Aided Secondary Analysis : consequences of such an Innovation for Sociologists in an Industrial Context. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.
- [143] Xin Li, Minghua Li, and Liren Zeng. Virtual classrooms supporting a two-way synchronized video and audio interaction. pages 446–455, August 2010.
- [144] Jimmy Lin, Rion Snow, and William Morgan. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 422–429. ACM, 2011.

- [145] K. Lyons and T. Starner. Mobile capture for wearable computer usability testing. In *Proceedings Fifth International Symposium on Wearable Computers*, pages 69–76. IEEE Comput. Soc, 2001.
- [146] Douglas Macbeth. On reflexivity in qualitative research : Two readings, and a third. *Qualitative inquiry*, 7(1):35–68, December 2001.
- [147] P P Maglio and C S Campbell. Tradeoffs in displaying peripheral information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 241–248. ACM, 2000.
- [148] Joe Marshall, Paul Harter, Jo Longhurst, Brendan Walker, Steve Benford, George Tomlinson, Stefan Rennick Egglestone, Stuart Reeves, Patrick Brundell, Paul Tennent, and Jo Cranwell. The gas mask. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*, page 127, New York, New York, USA, May 2011. ACM Press.
- [149] Ference Marton. Phenomenographydescribing conceptions of the world around us. *Instructional science*, 10(2):177–200, 1981.
- [150] Gregor McEwan, Carl Gutwin, Regan L. Mandryk, and Lennart Nacke. "i'm just here to play games": Social dynamics and sociality in an online game site. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW '12*, pages 549–558, New York, NY, USA, 2012. ACM.
- [151] D R McGee, P R Cohen, R M Wesson, and S Horman. Comparing paper and tangible, multimodal tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 407–414. ACM, 2002.
- [152] Iain McLeod, Huw Evans, Philip D Gray, and Rebecca Mancy. Instrumenting Bytecode for the Production of Usage Data. In Robert J K Jacob, Quentin Limbourg, and Jean Vanderdonckt, editors, *CADUI*, pages 183–194. Kluwer, 2004.
- [153] D McNeill. *Hand and mind: What gestures reveal about thought*. University of Chicago Press, 1992.
- [154] David McNeill. So you think gestures are nonverbal? *Psychological Review*, 92(3):350–371, 1985.

- [155] Florian Michahelles, Ramon Wicki, and Bernt Schiele. Less contact: Heart-rate detection without even touching the user. In *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, volume 1, pages 4–7. IEEE, 2004.
- [156] Pejman Mirza-Babaei, Lennart E. Nacke, John Gregory, Nick Collins, and Geraldine Fitzpatrick. How does it play better?: Exploring user testing and biometric storyboards in games user research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1499–1508, New York, NY, USA, 2013. ACM.
- [157] T P Moran, W Van Melle, and P Chiu. Tailorable domain objects as meeting tools for an electronic whiteboard. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 295–304. ACM, 1998.
- [158] Alistair Morrison, Paul Tennent, John Williamson, and Matthew Chalmers. Using Location, Bearing and Motion Data to Filter Video and System Logs. *Pervasive Computing*, pages 109–126, 2007.
- [159] Alistair Morrison, Paul Tennent, John Williamson, and Matthew Chalmers. Using location, bearing and motion data to filter video and system logs. pages 109–126, May 2007.
- [160] Cathy Murray. Secondary analysis of qualitative interviews: using NVivo to avoid the pitfalls of primary analysis. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.
- [161] E D Mynatt, J Rowan, S Craighill, and A Jacobs. Digital family portraits: supporting peace of mind for extended family members. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 333–340. ACM, 2001.
- [162] Christian Monrad Nielsen, Michael Overgaard, Michael Bach Pedersen, Jan Stage, and Sigge Stenild. It’s worth the hassle!: the added value of evaluating the usability of mobile systems in the field. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 272–280, New York, NY, USA, 2006. ACM.
- [163] D Norman. Some observations on mentai modeis. *Mental models*, 7, 1983.

- [164] Donald A Norman. *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex and Information Appliances Are the Solution*. MIT Press, 1998.
- [165] N Nova, F Girardin, G Molinari, and P Dillenbourg. The Underwhelming Effects of Automatic Location-Awareness on Collaboration in a Pervasive Game. In *International Conference on Cooperative Systems Design (COOP 2006)*, pages 224–238, 2006.
- [166] Hidehiko Okado and Toshiyuki Asahi. GUITESTER : A Log-Based Usability Testing Tool for Graphical User Interfaces. *IEICE transactions on information and systems*, 82(6):1030–1041, 19990625.
- [167] Eva Oliveira, Mitchel Benovoy, Nuno Ribeiro, and Teresa Chambe. Towards emotional interaction: Using movies to automatically learn users’ emotional states. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part I, INTERACT’11*, pages 152–161, Berlin, Heidelberg, 2011. Springer-Verlag.
- [168] A Oulasvirta and T Nyysönen. Flexible hardware configurations for studying mobile usability. *Journal of Usability Studies*, 4(2):93–105, 2009.
- [169] S Oviatt. Multimodal interfaces for dynamic interactive maps. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 95–102. ACM, 1996.
- [170] S Oviatt. Multimodal system processing in mobile environments. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 21–30. ACM, 2000.
- [171] Roy Pea, Michael Mills, Joseph Rosen, Kenneth Dauber, Wolfgang Effelsberg, and Eric Hoffert. The Diver Project: Interactive Digital Video Repurposing. *IEEE MultiMedia*, 11(1):54–61, 2004.
- [172] Shirlina Po, Steve Howard, Frank Vetere, and Mikael B Skov. Heuristic Evaluation and Mobile Usability: Bridging the Realism Gap. *Mobile Human-Computer Interaction {â}??MobileHCI 2004*, pages 49–60, 2004.
- [173] Erika Shehan Poole, Andrew D. Miller, Yan Xu, Elsa Eiriksdottir, Richard Catrambone, and Elizabeth D. Mynatt. The place for ubiquitous computing in schools. In

- Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 395, New York, New York, USA, September 2011. ACM Press.
- [174] Reid Friedhorsky, Jilin Chen, Shyong Tony K Lam, Katherine Panciera, Loren Terveen, and John Riedl. Creating, destroying, and restoring value in wikipedia. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 259–268. ACM, 2007.
- [175] J Prosser, editor. *Image Based Research: A Sourcebook for Qualitative Researchers*. Falmer Press, 1998.
- [176] M. Raento, A. Oulasvirta, and N. Eagle. Smartphones: An Emerging Tool for Social Scientists. *Sociological Methods & Research*, 37(3):426–454, February 2009.
- [177] B Rebsamen, E Burdet, C Guan, Haihong Zhang, Chee Leong Teo, Qiang Zeng, M Ang, and C Laugier. A Brain-Controlled Wheelchair Based on P300 and Path Guidance. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 1101–1106, 2006.
- [178] Kasim Rehman, Frank Stajano, and George Coulouris. Visually Interactive Location-Aware Computing. In Michael Beigl, Stephen Intille, Jun Rekimoto, and Hideyuki Tokuda, editors, *UbiComp 2005: Ubiquitous Computing*, volume 3660 of *Lecture Notes in Computer Science*, pages 177–194. Springer Berlin Heidelberg, 2005.
- [179] Peter Reichl, Peter Froehlich, Lynne Baillie, Raimund Schatz, and Antitza Dantcheva. The LiLiPUT prototype: A wearable environment for user tests of mobile telecommunication applications. In *CHI '07 extended abstracts on Human factors in computing systems - CHI '07*, page 1833, New York, New York, USA, April 2007. ACM Press.
- [180] Greg Ross, Alistair Morrison, and Matthew Chalmers. Visualisation Techniques for Users and Designers of Layout Algorithms. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*, pages 579–586, Washington, DC, USA, 2005. IEEE Computer Society.
- [181] Dana Rotman, Jennifer Golbeck, and Jennifer Preece. The community is where the rapport is—on sense and structure in the youtube community. In *Proceedings of the fourth international conference on Communities and technologies*, pages 41–50. ACM, 2009.

- [182] Dana Rotman, Jennifer Preece, Yurong He, and Allison Druin. Extreme ethnography: Challenges for research in large scale online environments. In *Proceedings of the 2012 iConference*, iConference '12, pages 207–214, New York, NY, USA, 2012. ACM.
- [183] Maria Roussou, Armin Cremers, Dirk Schulz, Mark Moors, Elias Spirtounias, Mika Marianthi, Vassilis Savvaides, Alexandra Reitelman, Dimitrios Konstantios, Andromachi Katselaki, Panos Trahanias, George Giannoulis, George Kamarinos, Antonis Argyros, Dimitris Tsakiris, Pantelis Georgiadis, Wolfram Burgard, and Dirk Haehnel. Experiences from the use of a robotic avatar in a museum setting. In *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage - VAST '01*, page 153, New York, New York, USA, November 2001. ACM Press.
- [184] Gilbert Ryle. The Thinking Of Thoughts: What Is 'Le Penseur' Doing? *University of Saskatchewan University Lectures, no.18*, 1968.
- [185] Harvey Sacks. The baby cried. The mommy picked it up. In G Jefferson, editor, *Lectures on Conversation*, pages 236–242. Appleton-Century-Crofts, 1992.
- [186] Stefan Scherer, Nadir Weibel, Louis-Philippe Morency, and Sharon Oviatt. Multi-modal prediction of expertise and leadership in learning groups. In *Proceedings of the 1st International Workshop on Multimodal Learning Analytics*, MLA '12, pages 1:1–1:8, New York, NY, USA, 2012. ACM.
- [187] Dominik Schmidt, Corina Sas, and Hans Gellersen. Personal clipboards for individual copy-and-paste on shared multi-user surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3335–3344, New York, NY, USA, 2013. ACM.
- [188] Holger Schnadelbach, Stefan Rennick Egglestone, Stuart Reeves, Steve Benford, and Brendan Walker. Performing thrill: Designing telemetry systems and spectator interfaces for amusement rides. In *SIGCHI Conference on Human Factors in Computing Systems (CHI2008)*. ACM, 2008.
- [189] J. Scholtz and S. Consolvo. Toward a framework for evaluating ubiquitous computing applications. *IEEE Pervasive Computing*, 3(2):82–88, April 2004.
- [190] Marc Schroder, Elisabetta Bevacqua, Roddy Cowie, Florian Eyben, Hatice Gunes, Dirk Heylen, Mark ter Maat, Gary McKeown, Sathish Pammi, Maja Pantic, et al.

- Building autonomous sensitive artificial listeners. *Affective Computing, IEEE Transactions on*, 3(2):165–183, 2012.
- [191] Rudy Schusteritsch, Carolyn Y. Wei, and Mark LaRosa. Towards the perfect infrastructure for usability testing on mobile devices. In *CHI '07 extended abstracts on Human factors in computing systems - CHI '07*, page 1839, New York, New York, USA, April 2007. ACM Press.
- [192] M Scott. Comparing corpora and identifying key words, collocations, and frequency distributions through the WordSmith Tools suite of computer programs. In M Ghadessy, A Henry, and R L Roseberry, editors, *Small corpus studies and ELT: theory and practice*, pages 47–67. Benjamins, 2001.
- [193] Julian Seifert, Adalberto Simeone, Dominik Schmidt, Paul Holleis, Christian Reinartz, Matthias Wagner, Hans Gellersen, and Enrico Rukzio. Mobisurf: Improving co-located collaboration through integrating mobile devices and interactive surfaces. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces, ITS '12*, pages 51–60, New York, NY, USA, 2012. ACM.
- [194] Claude Shannon. A Mathematical Theory of Communication. *Bell Sys. Tech. J.*, 27:379–423,623–656, 1948.
- [195] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction Design: Beyond Human Computer Interaction*. Wiley, March 2007.
- [196] W Sharrock and R Anderson. Epistemology. In G Button, editor, *Ethnomethodology and the Human Sciences*, pages 51–76. Cambridge University Press, 1991.
- [197] E Shih. The home of the future: An ethnographic study of new information technologies in the home. *Advances in consumer research*, 28:88–97, 2001.
- [198] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.
- [199] Petr Slovák, Joris Janssen, and Geraldine Fitzpatrick. Understanding heart rate sharing: towards unpacking physiosocial space. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, page 859, New York, New York, USA, 2012. ACM Press.



- [200] Ian Smith, Sunny Consolvo, and Anthony LaMarca. The Drop. *Computers in Entertainment*, 3(3):4, July 2005.
- [201] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [202] Tadeusz Stach, T. C. Nicholas Graham, Jeffrey Yim, and Ryan E. Rhodes. Heart rate control of exercise video games. pages 125–132, May 2009.
- [203] Susan Leigh Star and James R Griesemer. Institutional ecology, translations’ and boundary objects: Amateurs and professionals in berkeley’s museum of vertebrate zoology, 1907-39. *Social studies of science*, 19(3):387–420, 1989.
- [204] Lucy A Suchman. *Plans and situated actions: the problem of human-machine communication*. Cambridge University Press, 1987.
- [205] Michael Sühling, Christian Jansen, Muthuvel Arigovindan, Peter Buser, Stephan Marsch, Michael Unser, and Patrick Hunziker. Multiscale motion mapping a novel computer vision technique for quantitative, objective echocardiographic motion measurement independent of doppler: First clinical description and validation. *Circulation*, 110(19):3093–3099, 2004.
- [206] Paul Tennent, Stuart Reeves, Steve Benford, Brendan Walker, Joe Marshall, Patrick Brundell, Rupert Meese, and Paul Harter. The machine in the ghost: Augmenting broadcasting with biodata. In *CHI ’12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’12, pages 91–100, New York, NY, USA, 2012. ACM.
- [207] Paul Tennent, Duncan Rowland, Joe Marshall, Stefan Rennick-Egglestone, A Harrison, Z Jaime, B Walker, and Steve Benford. Breathalising games: understanding the potential of breath control in game interfaces. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, page 58. ACM, 2011.
- [208] Monica Tentori and Gillian R. Hayes. Designing for interaction immediacy to enhance social skills of children with autism. In *Proceedings of the 12th ACM international conference on Ubiquitous computing - Ubicomp ’10*, page 51, New York, New York, USA, September 2010. ACM Press.
- [209] Richard Uhlig, David Nagle, Trevor Mudge, Stuart Sechrest, and Joel Emer. Instruction fetching: coping with code bloat. In *ISCA ’95: Proceedings of the 22nd annual*

- international symposium on Computer architecture*, pages 345–356, New York, NY, USA, 1995. ACM.
- [210] Michel François Valstar. *Timing is everything: A spatio-temporal approach to the analysis of facial actions*. PhD thesis, Imperial College London, 2008.
- [211] M Van Dantzich, D Robbins, E Horvitz, and M Czerwinski. Scope: Providing awareness of multiple notifications at a glance. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 267–281. ACM, 2002.
- [212] Ana Belen Garca Varela, Hector del Castillo, Pilar Lacasa, and Sara Cortes. Analysing an educational project in the classroom using Transana: Children and parents learning together. In *CAQDAS 07 Conference: Advances in Qualitative Computing*, 2007.
- [213] Giasemi Vavoula, Mike Sharples, Paul Rudman, Julia Meek, and Peter Lonsdale. Myartspace: Design and evaluation of support for learning with multimedia phones between classrooms and museums. *Computers & Education*, 53(2):286–299, September 2009.
- [214] Sudha Verma, Sarah Vieweg, William J Corvey, Leysia Palen, James H Martin, Martha Palmer, Aaron Schram, and Kenneth Mark Anderson. Natural language processing to the rescue? extracting” situational awareness” tweets during mass emergency. In *ICWSM*, 2011.
- [215] Daniel Wagner, Thomas Pintaric, and Dieter Schmalstieg. The invisible train. In *ACM SIGGRAPH 2004 Emerging technologies on - SIGGRAPH ’04*, page 12, New York, New York, USA, August 2004. ACM Press.
- [216] Josh Wall. Demo i microsoft surface and the single view platform. In *Collaborative Technologies and Systems, 2009. CTS ’09. International Symposium on*, pages xxxi–xxxii, 2009.
- [217] R Want, B N Schilit, N I Adams, R Gold, K Petersen, D Goldberg, J R Ellis, and M Weiser. An overview of the {PARCTAB} ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–33, December 1995.
- [218] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The Active Badge Location System. Technical Report 92.1, Olivetti Research Ltd. ({ORL}), 24a Trumpington Street, Cambridge CB2 1QA, 1992.

- [219] Mark J. Weal, Danius T. Michaelides, Kevin R. Page, David C. De Roure, Mary Gobbi, Eloise Monger, and Fernando Martinez. Tracking and annotation in skills-based learning environments. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6. IEEE, March 2009.
- [220] Mark J Weal, Danius T Michaelides, Mark K Thompson, and David C DeRoure. The ambient wood journals: replaying the experience. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 20–27, New York, NY, USA, 2003. ACM.
- [221] Nadir Weibel, Shazia Ashfaq, Alan Calvitti, James D. Hollan, and Zia Agha. Multi-modal data analysis and visualization to study the usage of electronic health records. In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth '13*, pages 282–283, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [222] Nadir Weibel, Colleen Emmenegger, Jennifer Lyons, Ram Dixit, Linda L. Hill, and James D. Hollan. Interpreter-mediated physician-patient communication: Opportunities for multimodal healthcare interfaces. In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth '13*, pages 113–120, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [223] Nadir Weibel, Adam Fouse, Colleen Emmenegger, Whitney Friedman, Edwin Hutchins, and James Hollan. Digital pen and paper practices in observational research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12*, pages 1331–1340, New York, NY, USA, 2012. ACM.
- [224] Nadir Weibel, Adam Fouse, Colleen Emmenegger, Sara Kimmich, and Edwin Hutchins. Let’s look at the cockpit: Exploring mobile eye-tracking for observational research on the flight deck. In *Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA '12*, pages 107–114, New York, NY, USA, 2012. ACM.
- [225] Mark Weiser. The computer for the 21st century. pages 933–940, 1995.
- [226] J Wejchert. The Disappearing Computer, 2000.

- [227] Carol Wellington and Rebecca Ward. Using video to explore programming thinking among undergraduate students. *J. Comput. Sci. Coll.*, 25(3):149–155, January 2010.
- [228] M. Wetherell. Positioning and Interpretative Repertoires: Conversation Analysis and Post-Structuralism in Dialogue. *Discourse & Society*, 9(3):387–412, July 1998.
- [229] D Wieder and D Zimmerman. The diary: diary interview methods. *Urban Life*, 5(4):479–498, 1977.
- [230] R. Wolff, D.J. Roberts, and O. Otto. Collaboration around Shared Objects in Immersive Virtual Environments. In *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 206–209. IEEE, 2004.
- [231] Allison Woodruff, Margaret H Szymanski, Rebecca E Grinter, and Paul M Aoki. Practical strategies for integrating a conversation analyst in an iterative design process. In *DIS '02: Proceedings of the 4th conference on Designing interactive systems*, pages 255–264, New York, NY, USA, 2002. ACM.
- [232] Yan Xu, Xiang Cao, Abigail Sellen, Ralf Herbrich, and Thore Graepel. Sociable killers: Understanding social relationships in an online first-person shooter game. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, CSCW '11*, pages 197–206, New York, NY, USA, 2011. ACM.
- [233] Motoji Yamamoto. Real-time analog input device using breath pressure for the operation of powered wheelchair. In *ICRA '08*, pages 3914–3919. IEEE, 2008.
- [234] Dongsong Zhang and Boonlit Adipat. Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. *International Journal of Human-Computer Interaction*, 18(3):293–308, 2005.

# Appendix A

## Author's contribution to described software

### A.1 Replayer

- complete initial development of phase one - a dedicated playback tool for the Treasure game, featuring a complex animated locational visualization of system log data played with synchronized multiple video streams.
- Complete design and development of the distributed extensible architecture.
- Complete design and development of the server database system and internal selection management system.
- Complete design and development of all bridges to third party software: media bridges of OSX and Windows and the Google Earth Bridge
- Complete Design and development of RML and the Instrumentor system, as well as the RML parser component and generic parser API
- Complete design and development of the QCCI lo-fi synchronization tool
- Complete design and development of the simple stats tool
- Collaborative design and development of the Meta Tool visual query system.
- Worked in the team that developed the treasure game outlined in the Replayer case study.

All Graphical visualization components (Time Series, Event Series, Histogram, Correlation Tool, Mutual Information Tool) were developed by Dr Alistair Morrison at the University of Glasgow. He also collaborated with the author on the development of the visual query system for the Meta Tool

## **A.2 Digital Replay System**

- Adaptation of the system tool to enable cross platform functionality
- Complete overhaul of the user interface of virtually every aspect of the system
- Complete redesign and redevelopment of the Video viewer
- Complete design and implementation of the DRS Document viewer
- Complete design and development of the Image Viewer
- Partial redesign and redevelopment of the concordance tool
- Amalgamation of transcription editor and annotation table viewer tools
- Several log file processors and viewers made for specific projects not described in this thesis
- Integration of DRS with the SIDGRID qualitative research sharing framework at the University of Chicago

All other design and development of DRS has been by members of the DReSS development team, past and present: Chris Greenhalgh, Andrew French, Jan Humble, Mike Fraser and Stuart Reeves

## Appendix B

# Addendum

### B.1 Introduction

Since the initial submission of this thesis there has been significant change in the field of HCI evaluation. This chapter will demonstrate that while there may have been significant changes in the available data sources, in practice the central argument of the thesis remains unchanged: that is, that the challenge of supporting qualitative data analysis is not a methodological one, but rather one of exposing and representing new types of data in a way that turns them into accountable objects and makes them practically useful for qualitative research - indeed the appearance of more and more streams of interesting and usable data simply serves to strengthen the argument that there is a need for this ‘technomethodolical’ [38] approach to enabling its use.

In order to address this point the chapter will examine several key areas: First we will explore some examples of the new types of data streams that have become increasingly accessible over the last few years - in most cases these were data that were already available one way or another, but are becoming increasingly popular within HCI either because of the changing availability of sensors, or because of the subsuming by HCI of fields such as affective computing and social signal processing.

Of these new types of data we will initially examine social networking data - something touched lightly on in case study 3, but something which has become an increasing staple of qualitative research for various reasons including accessibility, uptake, public-awareness and the plethora of ‘new’ social networks which deal with specific media, such as *vine*<sup>1</sup>,

---

<sup>1</sup><http://www.vine.com>

*instagram*<sup>2</sup> and *foursquare*<sup>3</sup>, which deal with video, photographs and location respectively.

Next, we will look at the field of expression recognition. This is not in and of itself a new field - indeed it has a long and proud history, but the requirements for everyday use have up till recently been verging on the unfeasible. The difficulty of actually using it in real world situations is steadily reducing - there are more commercial tools, equipment is becoming cheaper or even consumer grade, and the computational hardware requirements are less (or rather consumer grade hardware is catching up with the requirements - for example CUDA is now a standard on most mid to high end consumer graphics cards), making it more accessible for use outside the pure challenge of simply achieving it and making it into a more realistic data type for use in qualitative analysis.

Finally for new types of data, we will explore so called ‘biodata’ by which we mean physiological data recorded from a participant’s body using sensors (or more accurately ‘biosensors’). Again this is an area that was touched on earlier in the thesis, in case study 1, but in a short period of time there has been an explosion of the popularity and subsequent use of such sensors in HCI. This can at least partially be attributed to the increasing popularity of the field of affective computing (and to a lesser extent social signal processing) which has all but merged into more traditional HCI. The representation requirements for handling and making biodata accountable are significantly higher than some other data types previously addressed such as location, since they invariably require some level of processing before they become practically usable - something we shall return to presently.

There are of course vast numbers of new types of data. The three examples above (social networks, expression recognition and biodata) have been chosen as representative examples of the kinds of new data suddenly and widely available to researchers, each of which presents specific challenges in representation before they can reasonably be applied when doing the business of analysis.

Having thus determined that there are many new data streams for the qualitative researcher to explore, we will next explore the treatment some of that data requires to make it practically usable for analysis - and look at data processing. This is an area that the original thesis left largely unexplored, beyond simple representation, but is something fundamentally required by many of these new data types.

Moving away from new types of data, and the requirements these create for use, the chapter will become more reflective, looking at how the main argument of this thesis stands

---

<sup>2</sup><http://www.instagram.com>

<sup>3</sup><https://foursquare.com/>



with respect to recently published work. First we will briefly look at a range of new tools which address the evaluation challenges in a similar way to Replayer and DRS - that is by combining and synchronising streams of different types of data. Then we will look at a number of representative examples of the successful use of this approach. We will the conversely look at several representative examples of studies which would have greatly benefited from following this technique - often contrasting them with studies which approached a similar subject, but used a synchronised data approach to better effect.

Next we will look at the practice of eliciting stories with data, including several examples and demonstrating that working through a data stream with a participant can be an excellent way to get participants to describe their experience, often in a level of detail that is not otherwise immediately accessible by interview.

At this point we will reflect on the continued validity of this approach and how it has been borne out in the evidence presented above. We will then revisit the challenges to evaluation of ubiquitous computing systems systems laid down in the related work chapter in the light of the contents of this supplement - focussing on the effect of these new types of data, and how our approach to evaluating ubicomp (or rather supporting the evaluation of ubicomp) can accommodate them.

Lastly we will discuss how all these threads cohere together to support the central argument of the thesis, demonstrating its continued relevance in the face of a changing field.

## B.2 New Types of Data

There has been a seismic shift in this field over the last couple of years. Not in the approaches taken to handling data per se, but rather in the accessibility of new (or actually often old) types of data streams. We will take as example three specific sources of data: social networking, expression analysis and physiological data, to demonstrate the new types of challenges associated with a growing cornucopia of data sources opening for designers, and subsequently evaluators, to understand and explore.

### B.2.1 Social networking data

Originally the thesis (particularly in case study 3) argued that social networking was a viable source for gathering ethnographic (and other qualitative research) data. This particular data source has proven to be methodologically challenging as outlined by [182], though

the conclusions of that paper imply that a process of selective application of established methods are necessary to perform ethnography in what the authors refer to as ‘large scale online environments’.

This data gives us an extraordinary picture of users’ behaviour in these online environments. However, perhaps due to the nature of the data source, in particular the size of the datasets, most of the analyses tend towards the quantitative. For example, [182] points out that we can study: “*Individually distinctive data*” identities, demographics, personal preferences, and contributed content, which can answer questions about *who* is online and *who does what* online; “*Structural data*” - showing ties and relationships, shared interests, zeitgeists, patterns of interaction and information dissemination, which describes who is *connected* to whom and how; and “*Activity logs*” detailing action and behaviour, such as search queries, navigation, reviews and favouring, which describes who does *what*, *when* and *where*.

Practically, these kinds of data are relatively easy to capture and seem to be a rich source for analysis, so they draw many researchers to focus on quantitative assessments of the structure of the networks and log analysis of activities. Several examples of well known studies follow this structural analysis approach e.g. [45, 136, 174]. But this type of data does not by itself not present the complete picture of what is happening in these large environments. One of the elements missing from such quantitative analyses is why people are doing what they are doing online in the first place.

It is here that ethnography and similar qualitative approaches can offer a great deal. There is much to be said for hybridising traditional forms of ethnographic analysis with a more computation oriented approach to analysis. Some approaches apply the method of social-network analysis e.g. [112, 181] while others make use of a method called “trace ethnography” [91] which requires collecting and assembling automatically generated traces of activity. This latter approach is akin to the system-log analysis discussed earlier in this thesis. Natural language processing has also been used to programatically track behaviour in online social networks e.g. [7, 120, 144, 214] which while not method in and of themselves, provide one possible computational route to accessing the data to be found within these online communities - perhaps necessary given the sheer volume of information. And that volume is extensive: According to [182] “*Current statistics estimate the number of weekly tweets at 1 billion, Facebook has more than 750 million active users, and 48 hours of videos are uploaded to YouTube every minute*”. And the methods we have discussed thus far

largely apply only to text-based online interaction - or in some cases to capturing content dissemination, while not addressing the content itself.

The increasing proliferation of ‘new’ social networks (Vine, Instagram, Foursquare etc.) which are based on a wider selection of data types (video, images and location respectively) as well as an increase in both uptake and accessibility in the more established social networks (Facebook, and particularly Twitter and YouTube) have provided a wealth of material that demands its own approach to understand and evaluate. This multimedia landscape demands support for tools and method able to support viewing the networks at a macro-level to understand the structure as well as a micro-level to understand the details of online behaviour. So the challenge becomes one, not necessarily of developing new methods, but of developing ways to explore these rich new datasets, which can be examined at various different granularities to tell us much about *what* is happening online, *who* is making it happen and *how* it is happening.

## B.2.2 Expression Recognition

One area within the wider field of social signal processing is that of facial expression analysis. This is a mature field in its own right and is capable of computationally telling us much about a person’s (presented) emotional state. One might question at this point what the benefit to qualitative analysis of such an approach might be? We posit here that it is two-fold. One is the process of automatic code generation (for example we wish to count how many times a user smiles in a video). Of course we could do this by hand, but such work is time consuming and requires little skill. Arguably automating such a process would allow for better use of a researcher’s time. Second is the idea of emotional leakage - that is micro expressions almost too small for a human to pick up. Computerised analysis of individual facial muscles *can* pick up on these cues and potentially provide valuable insight about the emotional undercurrents of a given interaction [73, 210].

During interpersonal communication we implicitly transmit cues that are indicative of our internal affective state. These cues may be spontaneous (beyond self-control) responses to encountered stimuli e.g. being startled, receiving a gift or receiving affection, etc. However, these cues can be posed, that is, deliberately invoked to mask a true inner feeling. Here we will define the terms for these posed and spontaneous facial expressions as *Macro* and *Micro* expressions.

- Macro Expressions - Archetypal facial expressions synonymous with prototypic emo-

tions. These include Ekman's 6 basic emotions [70]: disgust, happiness, sadness, anger, fear and surprise (see figure B.1).

- Micro Expressions - Facial muscle activity that is of a much finer granularity over that of prototypic facial expressions. These are thought to *leak* [71] feelings of affect through unconscious facial muscle activity



Figure B.1: Ekman's 6 basic emotions

There have been many attempts at rationalising about the activation of facial muscles, their relationship to a person's feelings, what information the various polymorphic forms of the face offer to other people and how these actions accompany our roles within society. Work in this area can have its origins traced back as far as Darwin and Aristotle. The most enduring of these attempts is Ekman's Facial Action Coding System (FACS) [73,210]. FACS is a sign and judgement based system that enables the encoding of almost all anatomically possible permutations of human facial neuromuscular activity. The system comprises of a set of Action Units (AUs) (There are 9 action units prescribed to the upper face, 18 to the lower face, and a remaining 5 actions that cannot be attributed exclusively to the upper or lower regions of the face) that describes a total of thirty two atomic <sup>4</sup> actions attributable to facial muscle activity.

FACS is formed on the basis that the face has a finite number of muscles, and each muscle has a finite number of actions it may perform. These muscle activities are of a naturally temporal nature so can only ever be in a single state at any given time. This implies the human face is, on some level, an encodable construct and that facial expressions can be modelled, categorised and stored computationally. This requires actions being conducted by a given muscle within the face to be accurately detected and compared against a robust database of existing learned expressions for an accurate determination of what that expression may rep-

<sup>4</sup>These actions are not divisible, that is they happen entirely independently but may contribute to more than one single facial expression

resent. Such databases include SEMAINE<sup>5</sup>, MMI Facial Expression<sup>6</sup> and GEMEP-FERA<sup>7</sup>. Built on top of FACS itself is emFACS [72] which connects given collections of facial action units to specific emotions. The emFACS framework can be used to detect particular expression activity as for example in figure B.2. It is important to realise here that what is being detected is “expressed emotion” rather than necessarily “true emotion”. While [72] makes some claims about the connection of the activation of particular action units to emotions (including those described as *emotional leakage*. Emfacs databases (and thus the training sets used to train most systems) are largely built on *acted* emotions. This work area should be carefully understood as *expression recognition* rather than *emotion recognition* - that is we may detect that a person is *expressing* fear or happiness, but not necessarily that they are *experiencing* it.

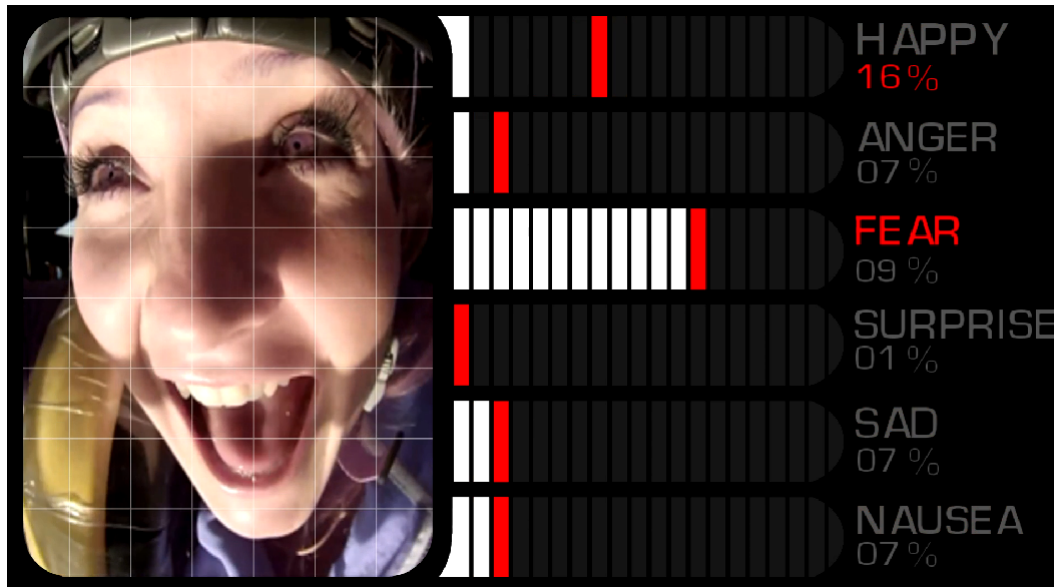


Figure B.2: Detecting facial expressions on a rollercoaster

This recognition method may offer an insight into the psychological, affective and cognitive states of other people and be subsequently applicable as a relevant data stream for deep analysis, either by automated coding or as an approach to (potentially) detecting emotional leakage.

<sup>5</sup><http://semaine-db.eu/>

<sup>6</sup><http://www.mmifacedb.com/>

<sup>7</sup><http://gemep-db.sspnet.eu/>

### B.2.3 Physiological data

The thesis, in case study 1, alludes to the use of recorded physiological data - sometimes called ‘biodata’ as one practically useful channel of information. As HCI has subsumed the respective fields of affective computing and social signal processing, biodata is becoming a far more prevalent data stream for a number of reasons. First, let us look at the accessibility of sensors. There are essentially three grades of physiological monitoring equipment:

- Medical Grade - Very expensive, designed primarily for use in labs, though some mobile equipment is available such as CamNtech’s *actiwave and actiwave cardio*<sup>8</sup> or compumedic’s *ambulatory EEG*<sup>9</sup>. Somewhere between lab-oriented and mobile are equipment such as Sten’s *nexus*<sup>10</sup> range - a medically rated multi sensor platform.
- Consumer grade - Cheap, easily available and becoming increasingly popular. There are plenty of examples to choose from but to pick a couple for representative purposes: *Polar’s* various heart rate monitors<sup>11</sup> and *Fitbit*<sup>12</sup>. Both of which connect to a users’ smartphone or PC and give them data about their exercise behaviour - generally in coordination with some proprietary application.
- Intermediate Grade - These are generally significantly more expensive than consumer grade equipment, but are not medically rated and thus more affordable than the medical grade equipment. They tend to be more lightweight and mobile and are popular for academic use. Examples include the *Affectiva Q Sensor*<sup>13</sup>, the *Empatica*<sup>14</sup> and the *Basis*<sup>15</sup>. These intermediate sensors are trending towards focussing on ‘wellness’ but typically provide good mobile access to electro dermal activity and heart rate sensing amongst other streams. There are also intermediate grade “sensor platforms” such as the *vilistus*<sup>16</sup> though thus far in practice these perform inconsistently when compared to their more expensive medical grade equivalents.

Another reason for the increasing uptake relates to the accessibility of method - that is, the fields of psychophysiology, social psychology and social neuroscience are all becoming increasingly visible within the HCI community. Measuring participants’ responses to interfaces

---

<sup>8</sup><http://www.camntech.com>

<sup>9</sup><http://www.compumedics.com>

<sup>10</sup><http://stens-biofeedback.com>

<sup>11</sup><http://www.polar.com>

<sup>12</sup><http://www.fitbit.com>

<sup>13</sup><http://www.affectiva.com>

<sup>14</sup><http://www.empatica.com>

<sup>15</sup><http://www.mybasis.com>

<sup>16</sup><http://www.vilistus.com>

- indeed controlling those interfaces with participants' physiological responses is becoming a common practice in HCI research - as example a search of the ACM digital library with the keywords "physiological data" returns some 1500 papers, and "psychophysiology" alone returns more than 50. For some specific examples see e.g. [61, 148, 177, 199, 202, 207]

It seems appropriate at this point to look at the types of biodata that can be measured. What follows is by no means an exhaustive list, but goes some way to show the range of biodata that might be accessible (sometimes with very small unobtrusive sensors - or in some cases simply by the application of appropriately positioned cameras).

- Electrocardiograms (ECG), Blood Volume Pulse (BVP), Heart Rate Variance (HRV). These are all measures of the heart. In the case of ECG and BVP these are often processed into a heart rate - used primarily as a measure of physical arousal, while HRV (typically computed from ECG) is often used to measure emotional arousal. It is also technically feasible to measure heart activity using only a camera as outlined in [205] or by micro impulse radar as shown in [155].
- Electrodermal Activity (EDA) sometimes called Galvanic Skin Response (GSR), psychogalvanic reflex (PGR) or Skin Conductance (SC) is a measure of electrical conductivity on the skin, typically related to the sympathetic nervous system and used to measure emotional arousal. It consists of two parts *tonic* and *phasic* which show different characteristics.
- Electromyography (EMG) is used to measure muscle movement - measuring the the electrical potential generated by muscle cells. It can be used to detect minute fluctuations in muscles for detecting (for example) micro fluctuations in expression.
- Electrooculography (EOG) and electroretinography (ERG) are used to measure, respectively, eye movement and retinal response to stimuli. These are often implemented as part of the eye-tracker systems which are familiar to lab based HCI experimentalists.
- Respiration (RSP) measured typically either by a chest-worn expansion sensor or a flow meter, though occasionally by a microphone is a measure of breathing and can be used to detect certain affective states such as fright, surprise, shock etc. based on the pattern of the breathing.
- Electroencephalography (EEG) and functional magnetic resonance imaging (fMRI) are used to study brain activity. EEG does this by measuring changes in electrical potential across the skull (correlated with activity in a particular area of the brain), while fMRI works on the principle of detecting changes in blood flow in the brain -

predicated on the fact that when an area of the brain activates, blood flow to that area similarly increases.

It should, from the above list, be clear that physiological data is a rich source of information, some of which (such as eye tracking) is already very familiar in lab-based HCI research and others which are becoming more practically viable for research ‘in the wild’. For a more in depth list of the types and relevance of different biodata, see [39]. While some forms of biodata such as fMRI might be impractical for mobile HCI/ubicomp there are plenty of examples where it is used to explore or control interfaces. When looking at these signals, which might seem dependent on extensive medical knowledge, it is possible to extract meaning from them by a process of abstraction and representation - using tools to turn a raw signal into something both meaningful and understandable. Handling biodata typically requires some work to transform it into usable data. This is of course no different from any other data type (for example consider the process of ‘cleaning’ data described in chapter 2) - however the methods in this case are more computational. This necessary data processing is the topic of the next section.

## B.3 Data processing

We can see from the preceding section on new types of data that many of these data types are not inherently information carrying in their raw form. That is, of course they contain data but they require some work to transform them into *accountable information*. This fits with one of the core arguments of the thesis - that data (for example transcriptions or system log data) need to be treated to be turned into accountable objects - that is, that the information contained in them needs to be teased out by manipulation, abstraction, representation etc. This is similarly true of the data described here. While some of that data, for example gathered tweets, text messages or written status updates, might be treatable using the same approach as described for transcriptions - albeit potentially on a somewhat grander scale, other forms of data require a more computational approach to be made useful. Handling biodata, for example, requires a specific technical approach in order to transform it into usable data. consider a captured ECG wave. It might be impractical to work with in its raw representation - assuming we consider the time series representation of blood flow through the heart to be a raw representation, but when processed and turned into a heart rate it may be considered usable. A time series of heart rate is a readable and relatively familiar model of



arousal. Of course if we wish to make comparisons between subjects we then have to address the issue of baselining and scaling, but strategies for approaching this are well defined and appear in publications such as the *Handbook of Psychophysiology* [39]. Similarly EDA in its raw form might be considered a useful measure of emotional arousal [39, 59, 114] however direct comparison between subjects (and even within subjects under different environmental conditions) is difficult. One strategy is baselining, as with heart rate, but EDA tends to drift [39] so modelling it as a rate of change ( $\Delta$ EDA) which still requires some scaling is desirable. This trend of necessary transformation is endemic in the use of biodata, but not necessarily problematical in itself since a number of tools such as *EDF Browser*<sup>17</sup>, *Vicarious* [206], *Biotrace*<sup>18</sup> and even more general tools like *Matlab*<sup>19</sup>, can help to support this process with accessible and descriptive interfaces.

Much biodata information is restricted to descriptions of arousal (whether physical or emotional) which may not tell us that much out of context. When combined with video (or some other form of social context carrying information) we can infer much. For example in case study 1 we looked at a participant riding a rollercoaster. The heart rate information there gives us an effective measure of arousal, while the context tells us what kind of arousal it might be - for that example excitement, fear, thrill etc. There may of course be other causes, but reviewing the video and looking for contextual clues either in facial expression or in some other information can help us put that arousal in context and explicate the participant's feelings. The data provides a practically useful way to find interesting areas of the video to unpack in greater detail - and knowing what that data tells us can be very useful. Sometimes this is familiar, as in the case of heart rate, and sometimes it requires some additional knowledge as in the case of EDA in social coordination - but if the unfamiliar data streams are turned into a familiar form (for example "arousal") - they become useful information carrying data streams in their own right.

Facial expression analysis similarly requires a computational approach to be useful. While there are various different approaches to capturing expression - the currently popular approach, based on facial muscle activation - and its counterpart schema FACS [73, 210] provides a set of activating 'facial action units'. These may be processed using FACS into a set of expressions, or by emFACS into a set of emotions (generally in each case what is actually created is a confidence value for each of several expressions or emotions, and is

<sup>17</sup><http://www.teuniz.net/edfbrowser/>

<sup>18</sup><http://www.humankarigar.com/biotrace.htm>

<sup>19</sup><http://www.mathworks.co.uk/products/matlab/>

achieved by a variety of machine-learning approaches). Once this processing is done, we have transformed the raw data stream (activating action units) into practically usable and understandable data - which can be used, for example, to automatically code video for particular expressions or emotions, or as suggested for an arousal stream - to locate key points of interest when working through a longer dataset - (e.g. a look of disgust when testing an interface might indicate a particularly poorly designed area which needs addressing).

Generalising then, it is practically feasible to collect so called invisible (or at least intangible) data such as biodata, or leakage in facial muscle activation, and transform that data into something understandable and usable in a deep qualitative analysis.

## B.4 New Systems

The purpose of the previous sections has been to demonstrate the range of data types becoming increasingly available, and to highlight the specific challenges that each of them present when attempting to use them for deep qualitative analysis. Here we look at how the method proposed by this thesis - of combining ‘born digital’ data with more traditional forms, has played out over the last couple of years. In particular we will examine a some newer systems developed in the interim, then in the proceeding sections we will look at a number of success stories, showing the application of this technique, then look at a representative sample of work that does not use this approach, to show how log files alone (or video/notes etc. alone) remain insufficient to meet this challenge - demonstrating that the central argument of this thesis remains intact.

### **Chronoviz**

*Chronoviz* [80], first developed in 2011 at the university of California, San Diego, provides a suite of tools similar in nature to digital replay system, in that it is designed to create synchronised representations of multi-modal data, including some support for time-based log files, and a ‘digital paper’ approach to capturing and synchronising field notes - indeed this is considered the main contribution of the work. In particular, *chronoviz* provides solid support for GPS data and numerical time series data.

### Cloudpad

*Cloudpad* [95], developed at the University of Nottingham (independently from the author), is a cloud-based documentation and archiving tool, that also serves as a multimedia annotation toolkit. It is built on the premise of collaborative analysis (in some ways similar to *Mixed media grid* [83]), and is hence delivered via a web interface. It allows for the synchronisation and annotation of multiple streams of video, audio and text, but also some light support for GPS tracking data.

### Vicarious

*Vicarious* [206], also developed at the university of Nottingham (in part by the author), is designed primarily as a live visualisation tool, aimed at synchronising visualisations of biodata and video data. Unlike most other similar tools, it provides functions for actively processing and transforming biodata to create understandable visualisations. While its use is principally performance-oriented, it can be used to create synchronised playbacks for analysis.

### CLAPS

*The combined log and audio playback system (CLAPS)* [100], developed at Texas A & M University, is designed specifically around audio analysis - in conjunction with log files. It focusses on following participant trajectories through experiences by means of either system log data, or recorded audio - allowing users to focus on extracting only relevant audio from longer term experiences. This focus on audio rather than video is unusual, but arguably appropriate for mobile ubicomp experiences when video is either difficult to film (i.e. participants are fast moving etc.) or overly invasive.

### Timestreams

*Timestreams* [29], while not dealing with video per se, is a platform developed to support the complex temporal access required when dealing with long-term data collection, such as environmental data. It is a plugin to wordpress, designed to allow for non-sequential, time specific, or looped playback of specific areas of recorded log data, and has been used by a number of artists (e.g. [119]) in the production of visualisation and (some) analysis of environmental data. While one might argue that this is not combining log data with video, as might appear to be the argument of this thesis, it is a tool that allows the transformation

of log data into (in the artistic example abstract) visualisations thereby making that log data into accountable and practically usable objects.

### Biotrace

*Biotrace*<sup>20</sup>, being the only commercial offering on this list, has recently significantly improved its “multimedia” support making it practically usable for combining recorded videos with biodata. It provides an extensive suite of tools for processing and visualising synchronised biodata and is heavily in use in experimental psychology/psychophysiology, though the principal purpose of the video component is to provide the stimulus for experiments. Nevertheless, recorded biodata is played back in synchrony with the stimulus (or external recording) which means it can be used to understand the context of the biosignals.

## B.5 Success stories

In this section we will review examples of recently published work which makes use of the combined-analysis approach outlined in this thesis, demonstrating the validity of this technique, and thus confirming the continued validity of the central argument of the thesis.

Weibel et al. in [224] demonstrate effective use of the combination of recorded logs of eye tracking and external videos in a deep analysis of cockpit behaviour. In particular they use their system to explore attentional behaviour when engaging in everyday task on the flightdeck - with the eye tracking providing a powerful resource for understanding exactly what each participant is attending to when combined with the context provided by the external videos. They demonstrate here that it is necessary for, non-interfering observational analysis (absolutely necessary in this safety-critical context) to be able to capture both the general context and the specific behaviour of the individuals.

Hollan, in [107] focusses specifically on the analysis of so called *activity trails*, that is recorded histories of interaction with a computer - using background-running software to capture extensive information about open programs, documents etc, with snippets of video. He demonstrates that this multimedia data-capture approach is a powerful tool for reconstructing interactional behaviour - particularly over extended periods of use, where the activity trail can allow the selection of relevant time frames from the data.

Fouse and Hollan [81] provide an interesting view of this approach, by focussing not

---

<sup>20</sup><http://www.humankarigar.com/biotrace.htm>

on the later stages of analysis, but on the initial pass-through stage. This is a somewhat reflexive approach that captures and visualises researchers' activity as they work through video data, essentially serving to show researchers which areas of a video were attention-grabbing, and supporting the process of multi-stage viewing. This may arguably be at odds with expectation, in that it encourages analysis based on a first look, rather than deeper analysis of a whole dataset, but in an age of increasing volume of video data to be examined, approaches like this may effectively help to cut down the datasets into more manageable chunks.

Weibel et al. in [223], focus specifically on the capture and synchronisation of pen and paper-based observational research, noting the importance of this in everyday observational practice, noting its use both in quantitative (i.e. tallies etc.) and qualitative (i.e. specific quotes or sentences) note taking approaches, performing a study with some 28 participants - all observational researchers, over an eighteen month period, and receiving generally positive feedback on the facility to combine synchronised notes with video etc - indeed demonstrating a behavioural evolution of note taking practice in light of the digital nature.

Scherer et al., in [186] use multimodal analysis of audio, video and writing, to explore leadership and expertise in learning groups. They make use of visualisation software (chronoviz) to synchronise, the various components of the data, and in particular make use of waveform analysis of the audio to explore vocal behaviour. This holistic approach to data collection and analysis is only made possible by the existence of such tools, and provides deep insight into methods to disambiguate leaders, experts and other students in group learning sessions, through detectable indicators.

In [222], Weibel et al. focus on exploring patient-physician communication when mediated by an interpreter from a multi-modal analysis point of view, in which an xbox kinect is used to capture gesture data, along with multiple angles of video. The paper explores the breakdown of communication patterns, particularly when dealing with complex technical (medical in this case) information. It also examines the use of artifacts (primarily paper in this case), allowing the synchronisation of the content of those artifacts to allow a deeper understanding of context during the analysis. The paper is aimed to show the opportunities for HCI to support the interaction - and is made possible by using HCI to perform a deep analysis of the interaction itself. A similar approach is used in [221], where the authors explore the usage of electronic health records, by automated gesture tracking with a kinect to generate codes for gesture, and audio analysis for speech to create interaction patterns

for analysis.

In [134] Kramer et al. set out to explore challenges associated with source code maintenance, and the usage of *call graphs* in this practice. The authors collected navigation events (essentially clicks in an IDE) as well as video of the user performing the task, then synchronised the log files of those events with the video data. The events were then manually coded for modifications of source code as well as text-searches, scrolling and other UI events. Crucially - the videos provided the context of specific events (not least whether they were successful) in a way that was practically much simpler to achieve than any automated method, then the output of the coding was used to drive an automated analysis model attempting to determine understanding (and ultimately maintenance task success) of the source code from IDE interactions. The use of external video in this process was crucial to understand the specific events and their meaning in a way which would simply not have been possible from the log-files alone.

Schmidt and Gellerson, in [187] explore the use of personal clipboards in shared multi-user surfaces, an interaction paradigm that is becoming increasingly common with the availability of hardware such as Microsoft's Surface Table [216] etc. In this work the authors consider the impact on interaction of different forms of clipboard to support personalised copy-paste behaviour in these multi-user situations. They consider three approaches: context menus, subareas and external handheld devices (smartphones). In each case user studies are carried out capturing both video and detailed system logs, and the interaction videos are coded using a combination of information from the system logs and field notes. These codes were then used in a detailed quantitative analysis of task-speed, and combined with more qualitative feedback from the users to construct a detailed discussion of the three different approaches.

Using a similar approach, Seifert et al., in [193] explore the integrated use of mobile devices and interactive, multi-user surfaces for collaborative tasks. In this case the authors outline the development of *MobiSurf* a system to allow collaborative tasks to be performed using a mix of mobile devices and shared surfaces, in which private information (such as passwords etc.) remain on the local device and public information is published to the shared device. This builds on similar work such as Doring et al.'s distributed poker game [66]. The researchers use a combination of video and system logs to effectively unpack the system in use.

In [100], Hamilton et al. discuss the use of a system that combines and synchronises audio with logged game events, and demonstrate that the combination of game events and audio is

an effective way of exploring users' interaction with a game environment. In particular, they allow a researcher to follow particular users through the experience by muting the additional audio tracks to focus on a specific individual's trajectory, bringing it live again when that user interacts with another. This is a particularly nice example because it eschews video, instead focussing exclusively on audio - which is less subject to occlusion, shakiness etc. being issues that plague mobile video capture - particularly in fast paced game experiences.

In [122] Kafai and Fields provide a very detailed analysis of player behaviour in the virtual world *Whyville*. They capture video, field notes and very detailed logs. The logs capture behaviour data from some 681 participants, while the video was recorded over two days at an after-school club and observed 21 participants. In the analysis, both the log data and the video provide deep understanding of the interaction behaviour, and in each case are used to support the conclusions of the other - in general the log data was used to establish the "who, what and where" of an interaction (bearing in mind that there are many more participants than just those being filmed), while the video was used to flesh out the context.

Christel et al. in [50] focus more on the end-users' use of synchronised video and textual data - demonstrating through a deep system log analysis of users' interactions with a library of over 900 hours of oral history videos, that there is a strong benefit for accessibility to be found in presenting fully synchronised and searchable text transcripts along with video in large databases - particularly in this field, in which the spoken content is crucial, but nuanced by the video - thus text alone is insufficient to deliver a complete experience.

## B.6 Counter Examples

In this section we will take a look at some representative examples of recent papers which may have benefited from the application of the research discussed in this thesis. While it is simply not feasible to produce an exhaustive taxonomy of such systems, the following cases serve to demonstrate the kinds of areas where this analysis approach could prove useful.

In [167], the authors aim to develop an emotional recognition system based on biofeedback. The stimuli in this case are movies. In each case, the movies were marked up by experts for their particular emotional expectation. Biodata was then captured from the eight participants, and used to train a state vector machine (SVM) based classification system. Output of the system was then compared to the expert-classifications of the movie. Where this system breaks down however, is in a failure to examine the specific participants'

behaviour. Emotion is highly subjective, and while it may be that the stimulus was designed to elicit a specific emotion, having video of the participants to explore in detail those cases where the classifier fails to match the expected emotion would have potentially led to a stronger analysis.

In another example, Xu et al. [232] use a mixed methods approach to analysing social interaction in first person shooters. They make use of the extensive server logs from Microsoft's Halo 3, then use these to inform semi-structured interviews with a number of participating gamers. What seems to be almost glaringly absent from the study is any analysis of games (and gamers) in play - and the actual interaction between the players. The interviews are about specific games, informed by general behaviour patterns, but data from the specific game discussed is not directly explored. Given the approach this would appear to be a logical step made possible by analysis of synchronised system log, screen recording and contextual video recording of the players' immediate behaviour - something exploited to great effect in [156].

In [77], Fern et al. discuss a process of mining 'old' HCI data to extract complex behaviour patterns about users. What is interesting is that they take very rich data (data that they mention often contains rich contextual information) and immediately abstract from that to a very high level. What they do not allow for is the reversal of that abstraction, which means that all that contextual richness is discarded. By maintaining the links between high and low levels of data analysis, it would be possible to drill down into the specifics of those complex behaviours in a way that simply is not possible with the approach taken. Interestingly, the purpose of the work is to determine high level strategies, but to do this they resort to abstraction from low-level data logs. The process is certainly powerful for getting an overall picture of user behaviour, but is severely limiting in terms of the detail - even though that detail is available in their data set.

In McEwan et al.'s work on board and card gaming websites [150] a particularly compelling aspect of this argument is revealed. The paper aims to explore the social dynamics of players in these online games, in particular looking at shared activity and verbal communication through extensive analysis of three months worth of server logs, however the authors seem oblivious to the social context of the actual player. It seems unreasonable to examine the online interaction in this context without also considering the situation of the user themselves. The authors refer to the need to accept game play as a "legitimate form of human interaction" as if to suggest that the participants have no other kind of interaction



engagement. This work would very much have benefited from examination of the social context of at least some of the players, as beautifully outlined in [122] above - since context and situation may significantly affect online behaviour - e.g. multiple people playing the same game in the same room (poker lan parties etc.), people playing while engaging in some other task (e.g. watching television), people playing alone and solely focussed on the game, or one of a host of other contexts. This qualitative aspect is necessary to put the quantitative analysis into perspective.

At the opposite end of the scale, studies such as [227] depend exclusively on video. In this example Wellington and Ward are examining how students learn to program. While the phenomenography-based [149] approach taken, may have been effective, it generated a tremendous amount of work, with each video requiring painstaking hand coding for the students' coding behaviour - specifically the mechanics of that behaviour. If we consider the work of Kramer et al. [134] discussed in the previous section, as comparison, we can see how unnecessary this work is - the application of system logs would have provided a rich "ready made" set of codes for much of this information, though of course a coding pass for indirect interaction (such as reading and re-reading) would still be required. Even this too could be achieved, by applying eye tracking (as in [224]) however there are extensive complexity overheads to this addition. The point remains however, that by exploiting simple logging processes, the authors could have achieved more for less expended effort. At a most basic level, the authors could have pulled the camera back to focus on several subjects at once, since log information could have been used to reconstruct the contents of the screens, while still maintaining sufficient resolution to capture the contextual and gesture behaviour (reading etc.). This would have allowed them to increase their subject count, since the experiment was conducted at a specific time in a class, and potentially further validate their results.

## B.7 Eliciting stories with data

One practical benefit to analysis provided by additional data streams such as those described above, is as a tool to elicit stories from experimental participants. Getting users to provide explicit and deep detail about their experience is a fundamentally challenging task in post-hoc interviews (and indeed in live 'think aloud' type scenarios). Providing them with data on a time-line around which to talk can support the participant by giving them a framework

to talk about. Interestingly, anecdotal evidence from a number of ‘in the wild’ experiences would suggest that more abstract concepts such as arousal or emotional trajectory actually prove more useful than more easily understandable data such as location. This is because showing a participant a peak in excitement (for example) will result in them describing the exciting parts of their experience (regardless of whether the peak actually indicates that particular part of the experience). As example, when chatting to roller coaster riders at an event at the Alton Towers theme park in the UK (recorded for the BBC’s Blue Peter in 2013 - see figure B.2), riders were shown an expression trajectory and asked to explain their feelings and actions on the ride. The responses were quite detailed; compare this to initial questioning without the data stream, where the responses tended to be much more general (e.g. “*It was brilliant - I was terrified the whole time*”). This particular example is one of several such which showed a trend of the data being a useful method of extracting stories. A similar approach was used in a recent advertising campaign for Nissan’s Juke car, wherein participants had their biodata recorded in a number of thrilling experiences (a skydiving simulator<sup>21</sup>, a track day drive<sup>22</sup> and an adventure trip to Morocco featuring quad bikes, dune buggies and a helicopter ride<sup>23</sup>). In each case the participant was ‘taken through their data’ and this was used to elicit detailed information about their experience - this was particularly relevant in the case of the Morocco event where four full days of data were captured and only some small parts of the experience were filmed. Of course in this case, the extraction of information was for broadcast interview rather than deep analysis, but the principle of the argument remains intact: Data is an attractive, and relatively lightweight way to get people to talk about their experiences.

This view has been recently borne out in the field of games user research in a paper from Mirza-Babaei et al. [156] in which players are taken through their experience using their physiological data in a *biometric storyboard*. This storyboard tracks their experience in the game, showing the physiological highs and lows (primarily arousal based). The user then describes their experience to a game designer (or evaluator) using the storyboard as a *boundary object* [203]. This approach supports users in reflecting on their emotional trajectory through the game experience, rather than the purely physical aspects of the experience, which are the more normal subject of such discussions.

---

<sup>21</sup><http://youtu.be/4dyHNSbvZtk>

<sup>22</sup><http://youtu.be/M9IT2sdIT4I>

<sup>23</sup><http://youtu.be/FAPiiu6S600>

## B.8 Reflection

We have seen over the sections of this addendum, that the work of evaluating ubicomp systems, and the work of using ubicomp systems to evaluate wider interaction are changing. New technologies like biosensors and expression recognition are becoming widely available, and we have explored some of the specific challenges involved in transforming these forms of data into accountable objects through linear transformations, machine learning, action coding etc. Even transformed into qualitatively valuable objects, they still present challenges to understanding, and often require the addition of contextual data to make them practically valuable in a qualitative analysis.

Alongside the explosion of new types of data, we have also seen the emergence of new types of environments, like massively complex social networks, which create never before seen challenges to evaluation, yet we have also seen the HCI community embracing this ([182]). These new forms and sources of data are being actively explored. New ways of looking at these data are being developed ([122,134]), new tools to handle combining them are being released ([80,100]) and older, established tools are being updated to handle the necessary combinations of data stream sources (biotrace). We have seen how looking at additional channels of data has supported evaluation through direct analysis ([107,224]), and as a boundary object for eliciting deep description of experience from users ([156]). We have also seen how the provision of additional streams of data can be of practical value to users themselves ([50]).

Conversely we have seen that failing to exploit these possibilities can weaken research outcomes ([77,167]), or at least make the process significantly more time consuming and difficult ([227]).

Ultimately this goes some way to validate the argument, that in the changing and increasingly complex space that is ubiquitous computing, it is necessary to embrace new types of data, new sources of data, and to find new ways to make use of that data in analysis. While ultimately the methods remain relatively consistent, the process of teasing out detail from the data must change - that is the tools change but the task remains essentially the same. This is axiomatic of tool development - the hammer and the nailgun may behave very differently, but they ultimately perform the same task, and both require skill to use correctly. Qualitative social science is well equipped to exploit *useful accountable objects* - the challenge to HCI is to be able to create, or at least allow for the extraction of those

*useful accountable objects* from the complex tangle of data created all around us every day.

### B.8.1 Revisiting the challenges again

In the related work and conclusion sections of this thesis a set of challenges associated with handling and evaluating mobile and ubiquitous data were presented. It is appropriate at this point to revisit these challenges taking into account what has been covered in this supplementary chapter.

#### Small Displays

The issue of small displays remains prevalent - indeed it is an issue which has only increased with the continued proliferation of smartphones, tablets etc. Further to this are a number of new types of data capture devices which have no display at all (for example fitbits, or empathica devices), but can be an integral part of an interaction process. The method of addressing this is largely the same as proposed in the body of the thesis. It is reasonable to expect to be able to reconstruct from system logs, the behaviour of a given system during an interaction. Tools like Replayer and DRS directly support this process.

#### Headphones

As above, the situation of audio-interaction has not changed significantly. Synchronised reconstruction of experiences from recordings and logs, as well as distribution of recording may be sufficient to support handling this particular interactional analysis challenge. However, there are now tools such as CLAPS [100], which focus specifically on audio to better support the process of extracting this demonstrably useful data stream.

#### Non-Collocation

Here we see a significant increase in the use of mobile social networking use. Facebook revealed in August 2013 that some 78 percent of its users make use of its various mobile device applications. Quite apart from the non-located interaction with social networks in the first place, this mobility puts an extra strain on methods of understanding interactional behaviour with and through social networks. We have explored here how a hybrid approach of network analysis, coupled with more traditional ethnographic analysis can help to understand behaviour in these online communities ([122]). The tools to support the observational

side of that depend on the same principles expounded by this thesis - distributed capture and reconstruction of interaction behaviour.

### **Invisible Interaction**

This supplementary chapter has highlighted many additional ways in which users may interact with (or be monitored by) invisible systems - data channels like biodata and expression analysis create unfamiliar streams of data that require treatment to be of practical value, in the same way as any other captured data - the difference being the more computationally, or signal-processing-oriented manner in which they must first be treated in order to become first class data to be used in deep analysis.

### **Distribution of Interaction**

We have seen here a variety of different examples of new channels through which users' may interact with systems - from new types of inter-connected social networking systems, through physiological capture (which may even serve to drive interaction in systems such as [148,207,233]), to expression or gesture based interaction with, for example, avatars such as those described in [10,190]. These new channels of interaction demand ways to capture and synchronise them with other forms of behaviour capture, and the tools described in this thesis provide exactly that facility - from data logging and visualisation to systematic reconstruction of context.

### **Interaction Time**

The issue of interaction time is something of crucial importance when understanding the nature of interaction with and through social networks. The non-immediate and asynchronous nature of the communication media used make it very difficult to understand exactly how people temporally interact with them - and this is something where the more quantitative approach to understanding social network interaction is useful, since it can often tell us *when* users interact with the network, both in terms of reading and in response. This can be used as part of a hybrid (or mixed methods) approach to categorise and characterise this interaction.

### Technological Breakdowns

This is not an area that has been particularly addressed by this supplement, but the increasing complexity of interaction media leads to this being an increasing issue. To briefly consider an example, in [206] we describe a system where participants were monitored with bluetooth based biodata capture devices (nexuses). The event was a one night performance displayed live to a cinema. On the night of the event, despite numerous successful tests throughout the day and preceding days, the bluetooth radio connection completely failed. It was later determined that this was due working within a busy nightclub environment - with lots of active bluetooth interference. It was necessary for us to simulate the biodata of the participants for the benefit of the live cinema audience. Because of the robust and flexible and dynamically reconfigurable nature of the software used (vicarious) we were able to seamlessly swap in the simulated biodata to replace the missing 'real' data, then later capture the 'real' data from the local recordings on the devices and reconstruct the event as it should have happened - or rather as it 'did' happen for the participants, but not for the audience. This is of course rather an extreme example, but it serves to demonstrate the need for redundancy and strategies to handle the seams of complex interactive systems.

### B.8.2 Conclusions

Over the course of this supplementary chapter we have considered the appearance of new sources of data available to qualitative researchers - discussing examples including social networking data, facial expression analysis data and physiological data. We have looked at the ways in which this data needs to be treated, processed and abstracted in order to be turned into accountable objects, in a similar, but often more computational manner to the ways in which system log data and transcription data has to be addressed as described elsewhere in the thesis.

We have examined how the work presented in this thesis fits in with current research practices, giving concrete examples of the practical benefits of applying the methods described, and contrasting that with examples where the application of this approach would have significantly improved work done using either log-files alone, or video analysis alone. We have also discussed some of the new tools developed since the initial version of this work which follow a similar approach. We have looked at one of the more obvious practical benefits for working with streams of data - eliciting stories from participants using data as

a cue. Finally, we have revisited the challenges laid out in the related work section (and originally revisited in the conclusion) to see how the new factors explored in this chapter relate to those challenges.

Throughout this chapter it has been made clear that the central argument of the thesis remains intact despite the extensive changes in availability of data. That is, that the challenges presented by the addition of new ‘born digital’ forms of data are not methodological per se, but rather of inclusion and representation. Qualitative research, and in particular ethnography, has always been effective at cherry picking interesting data, and turning it into information (or vice versa depending on choice of definitions of the terms) - take for example the adoption of photo and video ethnography and more recently an inclusion of sources like location as important. These new types of data require a higher degree of processing to turn them into accountable objects - that is, to create meaningful and practically usable representations and, as such, collaboration between computer science and social science is necessary to make the exploitation of these modes of data viable. HCI has a long interdisciplinary history, showing itself, as a field, quite capable of bridging the apparent gaps between social science, computer science and other relevant fields like psychology, social psychology, psychophysiology etc. However, for all this new data, the core work of doing ethnography, indeed of doing qualitative data analysis, has changed little. While the tools may be an ever shifting landscape, and to some extent the onus to approve, make use of, and understand these tools adds to the day to day work of a qualitative social researcher, ultimately they are just that: tools. If making the invisible visible is considered a worthwhile goal, and this thesis has argued that it absolutely is, then tools to address these new forms of data are necessary to support the way in which social researchers go about their daily work.