Un modèle pour la gestion et la capitalisation d'analyses de traces d'activités en interaction collaborative

Gregory Dyke

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 551 I

THÈSE

présentée par

Gregory DYKE

pour obtenir le grade de
Docteur de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Informatique

UN MODELE POUR LA GESTION ET LA CAPITALISATION D'ANALYSES DE TRACES D'ACTIVITES EN INTERACTION COLLABORATIVE

A MODEL FOR MANAGING AND CAPITALISING ON THE ANALYSES OF TRACES OF ACTIVITY IN COLLABORATIVE INTERACTION

soutenue à Saint-Etienne, le 7 décembre 2009

Membres du jury

Président :	Thierry CHANIER	Professeur des Universités, Université Blaise Pascal Clermont-Ferrand 2
Rapporteurs :	Daniel SUTHERS	Professor, University of Hawai'i at Manoa
	Alain MILLE	Professeur des Universités, Université Claude Bernard Lyon 1
Examinateurs :	Jean CARLETTA	Senior Research Fellow, University of Edinburgh
	Thierry CHANIER	Professeur des Universités, Université Blaise Pascal Clermont-Ferrand 2
Directeurs de thèse :	Kristine LUND	Ingénieur de Recherche CNRS, Université de Lyon
	Jean-Jacques GIRARDOT	Directeur de Recherche 2, Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialités doctorales :	**Responsables :**
SCIENCES ET GENIE DES MATERIAUX	J. DRIVER Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	A. VAUTRIN Professeur – Centre SMS
GENIE DES PROCEDES	G. THOMAS Professeur – Centre SPIN
SCIENCES DE LA TERRE	B. GUY Maître de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	J. BOURGOIS Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	E. TOUBOUL Ingénieur – Centre G2I
INFORMATIQUE	O. BOISSIER Professeur – Centre G2I
IMAGE, VISION, SIGNAL	JC. PINOLI Professeur – Centre CIS
GENIE INDUSTRIEL	P. BURLAT Professeur – Centre G2I
MICROELECTRONIQUE	Ph. COLLOT Professeur – Centre CMP

Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLANT	Didier	PR 0	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	MA	Génie Industriel	DF
BOURGOIS	Jacques	PR 0	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	DR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 0	Génie des Procédés	DF
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	IGM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 1	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	SMS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FEILLET	Dominique	PR 2	Génie Industriel	CMP
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	SMS
FRACZKIEWICZ	Anna	DR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
INAL	Karim	MR	Microélectronique	CMP
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LERICHE	Rodolphe	CR	Mécanique et Ingénierie	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Mécanique et Ingénierie	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 0	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences & Génie de l'Environnement	DF
THOMAS	Gérard	PR 0	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 0	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	MR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 0	Professeur classe exceptionnelle
PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR (DR1)	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
IGM	Ingénieur général des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Dernière mise à jour le : 22 juin 2009

To my father, who taught me wisdom,

my brother, who taught me patience,

and my mother, who taught me love.

Acknowledgements

As is customary with such work, I have had giants' shoulders on which to stand in order to accomplish it. I have been luckier than most and am deeply grateful to those people who helped me climb up there in the first place, stay up there when I felt dizzy, and to the giants with whom I have had the pleasure of discussing both their work and mine.

My thanks go out to my supervisors, Kristine Lund and Jean-Jacques Girardot for their friendship, for letting me choose my own path while preventing me from straying off into the thickets, and for being my compass when the twists in the path left me confusing north and south.

Thank you to Annie, Steven and Marie Line on whom I could always count for all the little things which actually take up vast amounts of time.

Thank you to the LEAD team and to all the other users of Tatiana who have shaped my understanding of analysis and who have very patiently put up with Tatiana's numerous real-world shortcomings. Thank you also to all the colleagues in other projects, and in particular to Lotfi and Madeth, my partners in crime for trace models.

Thank you to Jean Carletta, Thierry Chanier, Alain Mille and Dan Suthers for accepting to be on my jury. They are contributors to some of the main works cited in this dissertation and I feel honoured and privileged that they will be evaluating it.

A big thanks to Chris Teplovs for being so enthusiastic about the potential of my work and who now has a Tatiana of his own. I wish her all the happiness and joy that life can bring. Thanks also to Gilles and Cynthia, for music, dance, friendship and love and but for whose presence I would probably still be on page one. Thanks to my family and Cynthia's, for their pride, encouragement and support.

Last, I thank the reader who has made it this far in spite of my often convoluted metaphors. Although I am no giant, I hope my work will nevertheless enable you to see further.

Résumé

Dans cette thèse, nous nous adressons au problème de l'étude socio-cognitive d'interactions humaines, plus particulièrement dans le domaine de l'apprentissage collaboratif médiatisé par ordinateur. L'étude de telles situations passe par l'analyse de traces (fichiers de log, audio-vidéo, etc.) du processus d'activité et présente de nombreuses difficultés :

- Grande quantité de données multimodales complexes issues de sources différentes.
- Données fortement dépendantes du contexte dans lequel elles ont été produites.
- Variété des approches méthodologiques et épistémologiques d'analyse possibles.
- Difficulté de partage et réutilisation de données et d'analyses effectuées à partir de ces données.

Notre travail a porté sur une réduction de ces difficultés et nous présentons dans cette thèse nos trois résultats principaux. D'une part, nous proposons une description du processus d'analyse de ce genre données ainsi qu'un artefact générique permettant de recouvrir un grand nombre d'artefacts analytiques que nous avons pu observer et que nous nommons *rejouable*. D'autre part, nous présentons une étude et modélisation informatique des rejouables, et décrivons les quatre opérations fondamentales qui peuvent s'y appliquer : synchronisation, visualisation, transformation et enrichissement. Enfin, nous décrivons l'implémentation de cette modélisation dans un environnement d'aide à l'analyse par manipulation de rejouables que nous évaluons dans des situations de recherche réelles.

Tatiana (*Trace Analysis Tool for Interaction Analysts* – http://code.google.com/p/tatiana), l'environnement logiciel résultant, est basé sur ces quatre opérations fondamentales tout en intégrant de nombreuses possibilités d'extension de ces opérations pour s'adapter à de nouvelles formes d'analyse sans sortir du cadre de l'analyse au travers de rejouables. Cet outil est utilisé par plus de 10 équipes de chercheurs (France, Royaume Uni, Pays-Bas, Danemark, Hong Kong) sur des thématiques et objets d'étude variés.

Cette thématique de recherche s'introduit dans une problématique plus large de passage à l'échelle sur l'analyse de données d'interaction, de partage de corpus de données et d'analyses collaboratives et incrémentales sur ces corpus. L'objet que nous avons défini et implémenté permet à de nombreuses disciplines informatiques (recherche d'information, fouille de données, ingénierie des connaissances, interaction homme-machine, intelligence artificielle) de trouver un nouveau champ d'application dans l'analyse de traces, tout en tenant compte de la particularité de ces objets d'étude et des artefacts qui en sont issus.

Abstract

In this dissertation, we address the problem of the socio-cognitive study of human interaction, more particularly in the field of computer-supported collaborative learning (CSCL). The study of such situations can be performed through the analysis of traces (log files, audio, video, etc.) of the activity process, yet presents numerous difficulties:

- Large quantity of complex multimodal data from a variety of sources.
- Data which is strongly dependent on the context in which it has been produced.
- Variety of possible methodological and epistemological approaches to analysis.
- Difficulty in sharing and re-using data and analyses performed on this data.

Our work has been on the reduction of these difficulties and, in this dissertation, we present our three main results. On one hand, we propose a description of the process of analysis of such data, as well as a generic artefact which covers a large number of the analytic artefacts we have observed and which we call a *replayable*. On the other hand, we present a study and a modelling of replayables, and describe the four fundamental operations which can be applied to them: synchronisation, visualisation, transformation and enrichment. Finally, we describe the implementation of this model in an environment that assists analysis through the manipulation of replayables, which we evaluate in real-life research situations.

Tatiana (Trace Analysis Tool for Interaction Analysts – http://code.google.com/p/tatiana), the resulting software environment, is based on these four fundamental operations and integrates numerous possibilities for extending these operations to adapt to new kinds of analysis while staying within the analytic framework afforded by replayables. This tool is used by over 10 research teams (France, United Kingdom, the Netherlands, Denmark, Hong-Kong) to study a variety of phenomena.

This research theme addresses the wider issues of scaling up the analysis of interaction data and sharing corpora and analyses of these corpora to allow validation, replication and collaboration among analysts. The object we have defined opens up the applicative domain of trace analysis to many computer science disciplines (information retrieval, data mining, knowledge engineering, HCI, artificial intelligence), while simultaneously taking into account the particularity of these kinds of analyses and the artefacts which are used to perform them.

Table of contents

Table of figures

Introduction

The problem

Not so long ago, if you wanted a copy of a text, you would have had to write the copy yourself, or hire someone to do it for you. Now, texts can be photocopied or scanned and printed. Even with such technology, modifying a text is a question of using correction fluid and hoping that there is sufficient room to perform the modification without too much disruption. In certain cases, however, we have access to word processors or spreadsheets, which not only allow the modification of documents but are also able to propagate the impact of these changes: extra text can mean new page numbering, a different value in a spreadsheet can cause the sum of a column to be updated.

This is a story of designing computer support to create and edit the artefacts which we produce as a by-product or a goal of our work. Modelling such artefacts in digital form requires an understanding of the intrinsic structure, not only of individual documents, but of the class to which they belong (e.g. a textual document or a spreadsheet). It also requires understanding the operations which can be performed on them and how the result of these operations can be integrated. For various kinds of artefacts (newspapers, pictures, diagrams, books, league tables, musical notation, databases, 3d models, etc.), this story is at different stages of advancement. For some, such as books and newspapers, we have created powerful authoring and editing tools. In many cases, however, the end artefact is produced in a format which is not designed to be edited (pdf documents, newspapers and books are typical examples of this). For others, such as 3d models and musical notation, we are still struggling to provide good computer support for creating and editing artefacts in an integrated way.

For researchers in various fields who are trying to find ways of managing, analysing and interpreting their data, this story is a familiar one, with computers proving to be a blessing when they work and a curse when they don't. In this dissertation, we focus on researchers who are analysing human interaction, particularly in computer-mediated situations. The data to be analysed is often in the form of video and computer log files. Such recordings are reasonably well understood and the computer artefacts which represent them are now in widespread use. The only shadow is that of formats and compatibility, which is increasingly

not a problem for video (due to widespread use of video for many reasons and a vested interest for all parties in arriving at compatible solutions) and, while still a major headache for non-technical users (which of course is the majority of researchers), is only a minor problem for log files.

The artefacts which are created *during* analysis of such data are another matter. Aside from statistical data (for which spreadsheets and specialised statistical analysis packages provide a solution), such artefacts are not well understood and, more importantly, are frequently objects which can be authored, but which are very difficult to edit or use in an integrated way. Many such artefacts are produced as images or diagrams which explain the interaction. However, including additional data on a diagram, or propagating changes on a diagram back to the original data for further use are frequently as time-consuming as creating the original diagram (much in the same way as modifications of a pdf document are not carried back to the source). More generally, the situation is that described in Figure A: although data can be analysed by many tools (with some technical coaxing), the process is forward-only and the results of the analyses cannot be further analysed and are thus difficult to capitalise upon. In this dissertation, we examine the nature of analysis of computer-mediated situations, provide a model for some of the artefacts produced during analysis, showing how such artefacts can be used in an integrated way, and describe the software Tatiana, which implements this model.

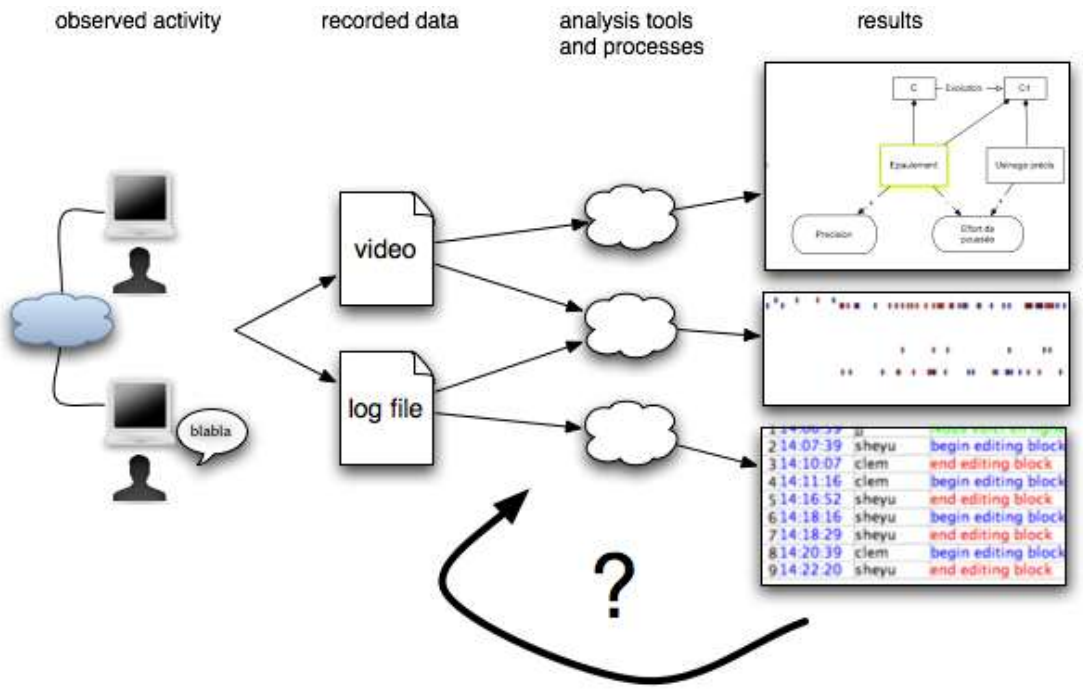


Figure A Recordings of an activity can be analysed by several different tools. The results of such analyses are difficult to feed back into subsequent analyses.

Research context

The work presented in this dissertation stems from a long-standing collaboration between the RIM laboratory (netwoRks, Informatics and Multimedia) at the Ecole des Mines de Saint-Etienne and the ICAR laboratory (Interactions, Corpora, leArning and Representations) in Lyon. This collaboration has been most notable in the European Commission funded projects SCALE[1] and LEAD[2]. In the first, RIM participated by providing a software platform to support collaborative argumentative learning. Data was collected from student's usage of this platform and analysed by ICAR (among others). Various analysis assistance tools were made during this project. The second project, LEAD, addressed networked communication in the classroom (i.e. mixture of face-to-face and computer-mediated communication) and included the development of communication software and an analysis tool specially designed for such data (the design and implementation of this tool formed the bulk of the work presented in this dissertation). At a local level, the Personalisation of EIAH (IT environments for human learning) project[3], has led to the discussion of the issue of traces (log files, or recordings) of human interaction: how they can be modelled and what purposes they can serve. Another local project, ASPIC[4], addresses collaborative design in engineering situations and has also been a source of experience for the analysis of collaborative interactions.

ICAR's experience in the analysis of CSCL data (computer supported collaborative learning – of which SCALE and LEAD provided examples) and collaborative interactions in general, combined with several projects within which research data and analysis practices have been shared have produced an ideal environment within which to address the problem of modelling and capitalising on analyses of traces of activity in collaborative interactions, with a more specific focus on CSCL data.

[1] Internet-based intelligent tool Support Collaborative Argumentation-based LEarning in secondary schools, IST-1999-10664, funded under the IST call of the 5th Framework Programme of the European Commission.

[2] Technology-enhanced learning and problem-solving discussions: Networked learning environments in the classroom, IST-2005-028027, funded under the IST call of the 6th Framework Programme of the European Commission.

[3] Funded by the Rhône-Alpes Region within the ISLE cluster.

[4] Funded by the Rhône-Alpes Region within the GOSPI cluster.

Dissertation outline

In Chapter 1, we give an overview of the field of CSCL and of the uses of traces for analysis and other purposes. We then present the state of the art in analysis tools and models and identify three shortcomings in the state of the art: the lack of knowledge about analysis, particularly concerning CSCL and how it differs from other kinds of analyses, models which do not clearly show what kinds of analysis problems they can address and tools which provide a set of functionalities without more clearly showing how they are integrated and how they constrain analysis to a limited set of practices. We position our work within the field of trace engineering and introduce the three main results of this dissertation, examining how they are related and how they will be evaluated.

In Chapter 2 we extend our exploration of the state of the art to the more general aspect of analysis, particularly in the field of CSCL. We use this to identify the specificities of CSCL analysis and to define the kind of analysis problems we are setting out to address: analyses of human interaction. More particularly, we organise the state of the art in analysis tools to examine the various kinds of objects created during analysis. One particular class of artefact, which we call a *replayable,* is the focus of our attention. We define this object through the operations which can be performed on it: synchronisation, visualisation, transformation and enrichment.

In Chapter 3, after a brief overview of the kinds of models and their aims, we propose a model for replayables which enables the operations defined on them. We discuss this model with regard to the state of the art and further examine the properties of replayables with regard to visualisation and transformation.

In Chapter 4, we present the analysis tool Tatiana (Trace Analysis Tool for Interaction Analysts), an implementation of the replayable model which provides analysts with an integrated analysis environment for the manipulation of replayables. We show how Tatiana has been designed with extensibility in mind, making it not only a platform for analysis but also an environment within which programmers can include new kinds of visualisation, enrichment and transformation and benefit from the existing operations on replayables.

In Chapter 5, we first present various case studies which have used Tatiana. We use these case studies to evaluate the results presented in the preceding chapters and discuss the limitations of our work. Finally, we examine the implications for future research in analysis of CSCL and

human interaction data in general, in trace engineering and in the use of computer science methods to automate the analysis of human interaction data.

1 Research Issues

1.1 Introduction

In order to define the research issues which need to be addressed in the development of a model for the analysis of CSCL data and for capitalisation on these analyses, we must first give an overview of the field and the work which is related to the problem at hand.

In this chapter, we will first examine the field of CSCL and some of the challenges faced during analysis. We will then give an overview of the usage of *traces* (a word to which we will later give a more precise definition but which can be roughly equated to recordings of human interactions) and present the state of the art in terms of methods for the analysis of these traces. Through discussion of the state of the art, we will show where there is a need for further research to be done resulting in a presentation of the three goals of this dissertation.

1.2 An overview of CSCL

The field of CSCL (or Computer Supported Collaborative Learning) concerns the study a) of learning in situations which are b) collaborative (or cooperative) and c) computer mediated. Koschmann (2002) defines this more precisely as "a field centrally concerned with meaning and practices of meaning-making in the context of joint activity and ways in which these practices are mediated through designed artefacts." (p.18) In this definition, the three core terms of "learning", "collaboration" and "computer support" can be elaborated upon.

Collaboration and cooperation have been distinguished (Dillenbourg, 1999) as being respectively an activity which is made with an effort to maintain a joint conception of a problem and an activity wherein roles are distributed and tasks are executed in parallel in a co-ordinated way. For example, moving a piano without damaging it must be done in collaboration by several people whereas moving a stack of boxes can be done in cooperation with each box moved by a single person.

Collaborative learning can be explained according to several epistemologies as summarized by Suthers (2006). Most accounts of learning during joint activity take a *constructive* stance. Constructivism (Piaget, 1976) suggests that learners learn through their own efforts at making

sense of the world. In CSCL, the emphasis is placed on intersubjective learning: that individual learning can occur through a group's efforts to create new meaning. This individual learning can be explained variously through a *knowledge-communication* epistemology (Wenger, 1987), through the claim that knowledge is jointly created by a group or, more extremely through a *knowledge building* epistemology, which extends the idea of *intentional learning* (Scardamalia & Bereiter, 1991) to group contexts: that groups of learners examine the limits of their knowledge and make a deliberate effort to extend this limit. Suthers suggests that CSCL should be centrally concerned with *intersubjective meaning-making*.

> *"To study the accomplishment (a post hoc judgment) of intersubjective learning we must necessarily study the practices (the activity itself) of intersubjective meaning-making: how people in groups make sense of situations and of each other." (Suthers, 2006, p. 321)*

In this context, computer-mediation can take various roles (De Vries, Lund, & Baker, 2002). It can serve as a collective memory of what has been constructed, allowing participants to review what has already been done, through the persistence of the medium. It can serve as a means of focusing the point of discourse and action, in order to better the *joint conception of the problem* necessary for collaboration or, in the case of learning situations which have been designed by a teacher, in order to constrain or guide the group activity. It can serve as a representational medium, within which participants can construct a shared artefact such as a text, an argumentative graph or a UML diagram. Finally, it can serve simply as a medium for communication. Suthers claims that

> *"the technology side of the CSCL agenda should focus on the design and study of fundamentally social technologies that are informed by the affordances and limitations of those technologies for mediating intersubjective meaning making." (Suthers 2006, p. 326)*

According to their epistemology of learning and their research questions, researchers may be led to design and evaluate pedagogical scenarios which can define a collaboration protocol (e.g. Jermann & Dillenbourg, 2003) or simply plan the work a group should do. Suthers (2006) argues that while the use of computer-mediation can help carry out such pedagogical scenarios, such activities could also be carried out with pen and paper and are not therefore central to CSCL. On the other hand, while the use of technology enables distant collaboration and/or learning, this is not a key characteristic of CSCL as all four roles of computer mediation can be valid in face-to-face uses of technology (although the need for computer-mediated communication when verbal communication is available is not immediately

apparent, but has been shown to be useful, for example, when anonymous discussion can be of benefit (Gelmini Hornsby *et al.*, 2008)).

Related fields to CSCL include CSCW (or computer supported collaborative work) and TEL (or technology enhanced learning). In CSCW, the context is a joint activity which is not concerned with learning, but with achieving a task. Some situations can be viewed as containing aspects of both these fields, for example when a teacher sets a task which must be carried out collaboratively, analysis could focus on how the joint activity is carried out or on how this activity is relevant to learning. In TEL, learning does not necessarily happen during a joint activity but is computer-mediated in some way. This can include many other epistemologies of learning, such as those which are more teacher centric (e.g. the teacher transfers knowledge to the students), and includes several roles for the computer, such as a communication device, a guide or a simulator.

1.2.1 Research questions in CSCL

Some examples of research questions in CSCL include identifying the link between a usage pattern, a tool or a pedagogical scenario on the one hand and a learning outcome or the quality of a learner's productions on the other (e.g. Lund, Molinari, Séjourné, & Baker, 2007), bringing about a given pattern whose outcome may be positive (e.g. Ronteltap, Goodyear, & Bartoluzzi, 2004), or studying how learners appropriate technological affordances (e.g. Overdijk & van Diggelen, 2008).

The methodologies to examine these questions are summarized by Suthers (2006) into three traditions: iterative design, experimental and descriptive. Iterative design tends to explore the "space" of technological affordances for learning, gradually improving technological artefacts by identifying what seems to be more or less favourable. Experimental studies compare the outcome of a group in a given condition with that of a group in a control condition, using methods such as content analysis (Strijbos, Martens, Prins, & Jochems, 2006) where codes are attributed to various events and statistical analysis is performed on those codes. One of the difficulties in this case is achieving sufficient intra- and inter-group homogeneity to achieve results with sufficient statistical significance. Another major difficulty is sufficiently isolating the variables of study so that the control condition differs from the experimental condition only in a few well understood ways. Finally, the necessity of controlling the conditions means that they are necessarily contrived situations whose results might not be applicable to more "ecological" situations. Descriptive studies rather attempt to describe what happened in

"natural situations", letting the interpretation emerge from the data, using methods born from ethnomethodology such as conversational analysis (Sacks, Schegloff, & Jefferson, 1974). These studies tend to be time-consuming and do not easily lend to predictive generalisations. Strijbos & Fischer (2007) describe how the limits of these various methods lead them to be combined either by triangulation of several results (mixed methods) or by developing new methodologies which are particularly adapted to a particular goal (hybrid methods). They explain that in order to do so successfully, a deep understanding of the constituting methodologies is necessary and that the standards to which the analysis has been held must be clearly spelled out in order that other researchers be able to understand and evaluate the results.

Whatever the methodology used, researchers are led to collect some kind of data and construct an analysis of that data. This data can include recordings of the activity which is being studied, pre- and post- tests and other information such as interviews, surveys, etc. Learning can then either be evaluated through tests or, as advised by Suthers (2006) and Koschmann (2002) by examining the activity itself and showing that it is composed of events which – according to a certain epistemology of collaborative learning – lead to learning.

In this dissertation, we are primarily concerned with the kinds of methodologies whereby the activity itself is examined – and which therefore have led to the recording of data about this activity. This focus stems from the kind of research which is carried out at ICAR and more generally within the LEAD project.

1.2.2 Difficulties in the analysis of CSCL data

In order to record a CSCL activity, it is first necessary to record the computer-mediated interactions (leading to log files or what some authors call a *digital trace*). However, this is often not enough, video being necessary to capture the relevant offline context (Avouris, *et al*., 2007). Harrer *et al.* (2007) draw on the collective experience of the researchers in the Cavicola project to define the analysis process as being composed of a sequence of operations including capture, segmentation, annotation and coding, analysis, visualisation and interpretation. These operations are difficult for a number or reasons, particularly when they are not assisted by technology.

First, the recorded data is often hard to understand from a syntactic and semantic standpoint. Recordings in the form of log files are typically opaque to a non-technological reader, being

written in a shorthand which is designed to be understood by computers rather than humans. Once the actual format is understood, the reader then has to reconstruct the activity which may have led to such a recording. In the best case, the events which were chosen to be recorded in the trace match the kind of events which are of interest to the researcher (e.g. turns taken in a chat dialogue). In other cases, however, the events are not at the correct kind of granularity or do not directly show the whole of what happened. For example, in a game where children must reconstruct a picture story by placing images from a scrambled pool in the correct order, the placing of the last image causes all incorrectly placed images to return to the pool. However, the events that are recorded only show which images were placed where and the reader must know how to fill in the gaps. The difficulty in understanding the log files is such that we have often observed[5] screen capture being used as a "catch all" method which, while less structurally rich (digital traces at least already include the segmentation of the activity into events) is easier to understand and ensures that nothing that happens on screen is omitted in the recording, despite the fact that such recordings take up large amounts of disk space and are more difficult to analyse automatically[6].

Second, the data can be difficult to understand from a pragmatic point of view. Particularly in the case of computer-mediated, face-to-face situations, the activity can be both multi-media and multi-modal (Lund, 2007) encompassing speech, gesture, pen and paper, computer-mediated communication and computer-mediated artefact construction. The recording of such an activity will be spread across video recordings and digital traces, both of which must be understood in concert in order to fully understand the activity (Goodman, Drury, Gaimari, Kurland, & Zarella, 2006). This poses two challenges:

1 ensuring that it is possible to synchronize the recordings (because the timestamps have been aligned or because at least one event occurs across the different recordings and serves as a synchronisation point)
2 performing the actual synchronisation between the two recordings during analysis.

In the case of multiple videos, the second is often achieved by creating a single video which combines these videos in a grid. This solution has the disadvantage of an increase in screen real-estate taken up by the new video and the loss of all but one of the audio channels (unless it is possible to mix the audio channels together in some way that is comprehensible).

[5] In our discussions with various researchers over the past three years.
[6] The ideal is, of course, to combine screen capture and log files.

In the case of video combined with digital traces there currently exists no generic solution, although this problem has been addressed by several tools (as we shall see when examining the state of the art).

A third challenge is posed by the sheer quantity of data. Depending on the methodology used, it is frequently unrealistic to analyse a complete corpus in detail. The question is then raised of how to get an overview of a situation and how to identify specific episodes which are appropriate for further investigation.

A fourth challenge arises when carrying out such steps as annotation, coding and visualisation. In such cases, it is frequently necessary to create representations which record the analytic work which has been carried out. Even with computer assistance, it is difficult to automatically create these representations and to create them in a way which renders them shareable and reusable. For example, consider two analyses (cf. Figure 1-1). The first consists of graphically representing the reply-structure of a discussion in a chat (i.e. the researcher transforms a chat log into a graph with boxes and arrows, showing a possible interpretation of the reply-structure underlying the chat). This would typically be recorded as an image, created by hand (or at the very least, the graph structure would, by necessity have to be created by hand). The second analysis consists of coding the same discussion according to the rainbow coding scheme (Baker, Andriessen, Lund, van Amelsvoort, & Quignard, 2007) whereby each utterance is coded according to its role in argumentation. This would also be performed by hand, most likely in software such as Excel™. Without dedicated software to support such an activity, there is no way to combine these two analyses, for example by attributing a colour to each code and representing these colours in the graph, without dedicated programmatic intervention (or tedious work by hand).

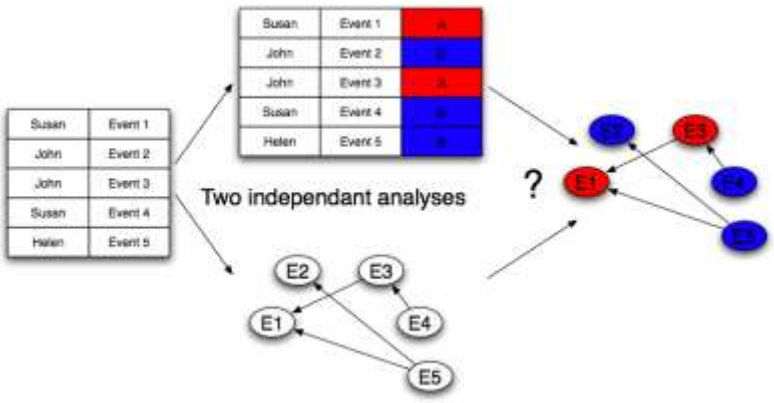


Figure 1-1 Given two independent analyses such as a colour coding and a contingency graph, how can the two easily be combined ?

These challenges are not all of the same nature. The first two are technical or technological, the third is methodological and the fourth and most important combines both these aspects and is at the core of the issues we address in this dissertation. They are an introductory overview of the difficulties currently faced by researchers wishing to analyse CSCL data.

1.2.3 Capitalising upon and sharing analyses

The last of the previous challenges is emphasized by the necessity of sharing and re-using corpora and analyses. Sharing and re-use is necessary for teamwork and can take several forms: spreading the workload across several team members (Goodman *et al.*, 2006), validating analyses through inter-coder reliability (De Wever, Schellens, Valcke, & Van Keer, 2006), extending the applicability of an analysis scheme to a new field (e.g. Lund, Prudhomme, & Cassier, 2007) or combining the insights of several analysts (e.g. Prudhomme, Pourroy, & Lund, 2007).

At a more community-wide level of consideration, in fields such as physics or chemistry, it is customary to share data when publishing in order that other researchers might verify the interpretation of that data. It is also customary to re-run experiments and perform the same analysis in order to replicate findings. In CSCL, this is not the case, despite recent promising efforts by Reffay, Chanier, Noras & Betbeder (2008), who have been examining how to structure learning corpora in order to share them.

1.2.4 Computer support for analysis

In spite of success stories in using computers to aid this analysis (e.g. Cox, 2007) and the existence of several tools to support it (cf. §1.4), these tools are frequently written to solve very specific problems and require further programmatic intervention to be adapted to new problem sets. Many authors have highlighted the need for better and more generic computer support for analysis (Fisher & Sanderson, 1996; Hilbert & Redmiles, 2000; Thomas & Cook, 2005; Suthers, 2006). Before examining the existing tools, we first need to define in greater detail the nature of recordings of human activity, particularly in computer-mediated situations.

1.3 Traces of human activity

In an ideal situation, a person wishing to study an activity would observe it (being therefore limited to what is *observable*) and analyse it directly, making sure to observe exactly that which is pertinent to their analysis. The fact that this is impossible in practice leads to the necessity of creating a *recording of the observable events which are considered pertinent with regard to some goal*. In this dissertation we will define a *trace* as being such a recording. This inherently poses a paradox (Georgeon, 2008): how can we know what is pertinent to record without having studied it? Does not the fact of judging something pertinent and therefore worthy of recording beg the question of what will be found pertinent during the analysis? Ericsson & Simon (1993) answer that "in designing our data-gathering schemes we make minimal essential theoretical commitments, then try to use the data to test stronger theories" (p. 274). The choice of tracing an activity must therefore in some way presume that the data which is needed to coherently analyse that activity is present in the trace.

Informally, the "traces" (or in English a more familiar word might be "tracks") of an activity, are the marks which that activity leaves on the environment (but which are not necessarily the goal of the activity and which are not necessarily left intentionally), which can in some way be interpreted in order to make sense of what happened and when. In this kind of situation, some traces are more durable and easier to interpret than others, depending on the medium into which they are inscribed (compare the tracks left by an animal in mud with those left in sand with those left in grass). To extend this metaphor, recording an activity is the act of intentionally manipulating the environment in which it occurs in order that the trace that is recorded be as appropriate as possible to the person who wishes to use it.

In many cases, the modification of the environment for tracing is simple (and potentially disruptive): placing a video or audio recording device in one or more strategic positions. Other modifications include the use of GPS receivers or eye-trackers to trace respectively the usage of gaze and location in the activity being observed.

1.3.1 Tracing computer-mediated activity

The tracing of computer-mediated activity is a special situation in that it is both possible to be very specific in what is traced, and to do so without modifying the environment in a disruptive way. The simplest way is through screen capture, but this does not fully exploit the

fundamental nature of computers – that all actions of the user on one of the computer's input devices *are known* to the computer, even if individual programs do not exploit them. By default, these actions do not leave a permanent trace. Tracing must therefore be an intentional action of the programmer, telling the computer to record certain specific events (Laflaquière & Prié, 2009). These might be user-interface events or higher level-ones. This kind of trace can be seen as the transcription, at various levels of granularity, of the screen recording which might otherwise be performed. These traces are sometimes referred to as *digital traces* (*ibid.*) but this term is confusing and should probably be avoided. Indeed, a video recording can be made digital, and a screen capture is certainly digital, but neither matches the term intended by the word "digital trace". We will discuss the properties of different kinds of traces in chapter 3, until then using the term *digital trace* to refer to an event-based trace of computer-mediated interactions.

In a collaborative computer-mediated activity, a single event might have different effects at different levels: clicking on a button (UI level), triggering a given action (local program level) and sending a message (distributed program level). As nothing is traced by accident, choosing what to trace becomes an important consideration. One answer could be "everything", but this can lead to huge amounts of data that we are as yet ill equipped to deal with. Another approach, used in the study of collaborative work, is to trace only those actions which are collaborative (both for the pragmatic reason that they often transit via the server in a server-client architecture and for the epistemological reason that). Harrer *et al.* (2004) define a collaborative action as:

> *An action that affects or can affect the collaborative process. The action itself or its effect should be perceived by at least a member of the group distinct of the one that performed the action. (p. 6)*

However, Mühlenbrock (2004) suggests that actions can present various degrees of interactivity, depending on parameters such as whether the action is performed in presence of others, whether it is related to other actions, etc. This indicates that whether an action is collaborative or not (and therefore might be worthy of being traced or not) is subjective and highly contextual. Other solutions to the problem of what to trace range from the pragmatic approach of tracing as much as possible and hoping that it will prove sufficient, to a very precise definition of what needs to be traced to answer a given research question (Laflaquière & Prié, 2009).

1.3.2 Uses of digital traces beyond analysis by researchers

Digital traces can be exploited within a wider context than analysis by researchers, in order to provide a better experience to users of digital environments. The treatments applied to traces in these situations differ from those applied by researchers in that they must often be automated (in order to be exploited in real-time), leading to research being performed on visualisation (Dimitracopoulou & Bruillard, 2006) and the calculation of *interaction indicators* (Harrer *et al*., 2004), defined as being the result of application of some data processing method and being somehow related to the mode, process or quality of interactions in a group setting.

The first "audience" for traces are the producers of these traces themselves. Traces can be used to present the user (or group) with some kind of feedback to his past activity (Jermann, Soller, & Mühlenbrock, 2001), ranging from a mirror to increase self-awareness, to a guide which attempts to steer the user towards some (presumably desirable) activity. Another use is to allow the system or program which produced the traces to automatically modify its behaviour, based on how it has been used in the past, adapting itself to the user and affording a personalised experience (Laflaquière, Champin, Prié, & Mille, 2005). Finally, the user can browse his own traces (such as web histories), using them as an inscription of his past activity rather like some people take notes of how they accomplished something or record where they have been and what they did (Laflaquière & Prié, 2009). In a group context, particularly in the workspace, this extends to an enterprise knowledge-base.

The other "audience" for traces are people who would like an overview of what a user, a set of users or a group is doing (e.g. France, Heraud, Marty, & Carron, 2007; May, George, & Prévôt, 2008; De Laat, Chamrada, & Wegerif, 2008). In learning situations, this typically concerns teachers and tutors who don't have the time or ability (in the case of distant learning) to interact with each student individually, but who want to know which students and groups are performing well and which are in need of assistance. Outside of learning situations, web analytics software allows webmasters to examine who is visiting their site, and network analysis software allows network administrators to monitor their systems, possibly looking for fraudulent activity (Denning, 1986).

1.4 State of the art in trace analysis

Many approaches, tools and methods exist to enable computer support for the analysis of traces. Ideally, we would have separated tools (usually described by their list of functionalities) from more elaborate (and re-usable) attempts at modelling traces and their analysis. In practice, some of the models have given birth to tools and some tools address a specific situation but have not been created with a broad model in mind. This presentation of the state of the art will be broadly broken up into categories: trace models, computer science methods for automated trace analysis, audio-video analysis, trace analysis tools, and exploratory sequential data analysis and replay tools.

1.4.1 Trace models

In this section, we examine various models which have been proposed to represent traces and analyses. In each case we will look at the motivation to model the trace, the approach that was used, any resulting tools and discuss the limits of this approach.

1.4.1.1 Cavicola

Motivation. Cavicola (Computer-based Analysis and Visualization of Collaborative Learning Activities) is an ERT (European Research Team) within the Kaleidoscope network (a European network of excellence for technology enhanced learning which ran from 2003 to 2008). This team is an institutionalisation of various partnerships between research laboratories in Patras, Valladolíd, Duisburg and Freiburg and continues work that started from an even larger consortium in 2001, centred around the concept of *interaction analysis*.

One of the core issues addressed by this consortium (Martínez *et al.*, 2005) is exemplified in Figure 1-2. When looking at different tools to analyse CSCL environments, they realised that each was created by a research team to answer their own specific questions regarding their own specific learning environment.

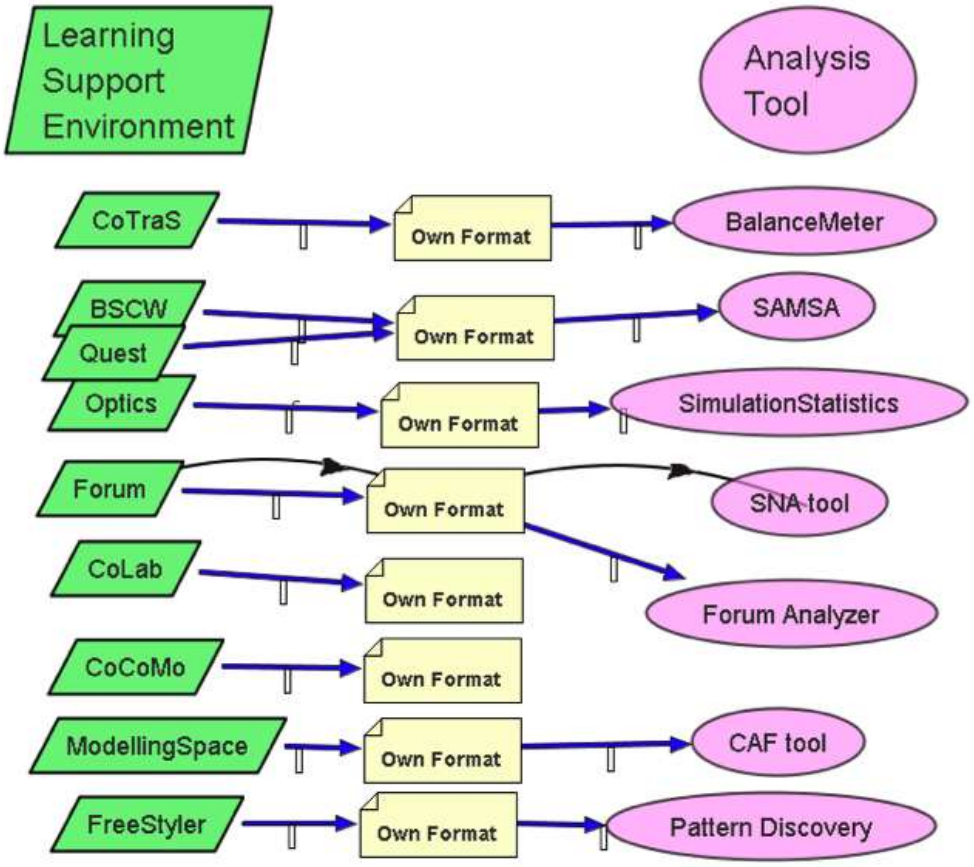


Figure 1-2 The initial situation: each tool is linked to a trace format and to the tool from which the trace is recorded (Martínez *et al.*, 2005, p. 13)

An accidental side-effect of this situation is that each trace producing environment and its corresponding analysis tool are linked to a particular file format, making it difficult for different analysis tools to be reused on data from other environments.

Approach. One of the goals of the consortium being to create a library of interaction analysis tools, they decided that they should also define a *common format* which would be understood by all the tools. Part of the XML DTD for this common format is presented in Figure 1-3.

This format describes interaction data (a digital trace) as being composed of a preamble describing the identifiable entities which can be found in the trace and a set of actions each of which has some required and some optional properties, such as the timestamp, the type of action, the user(s) involved and their role, the content describing the action and the objects on which the action applies. The goal of this format is to serve as an intermediary between trace producing tools and trace analysing tools which either directly produce and read this format or which conserve their own format, transforming between the two via XSLT transformations.

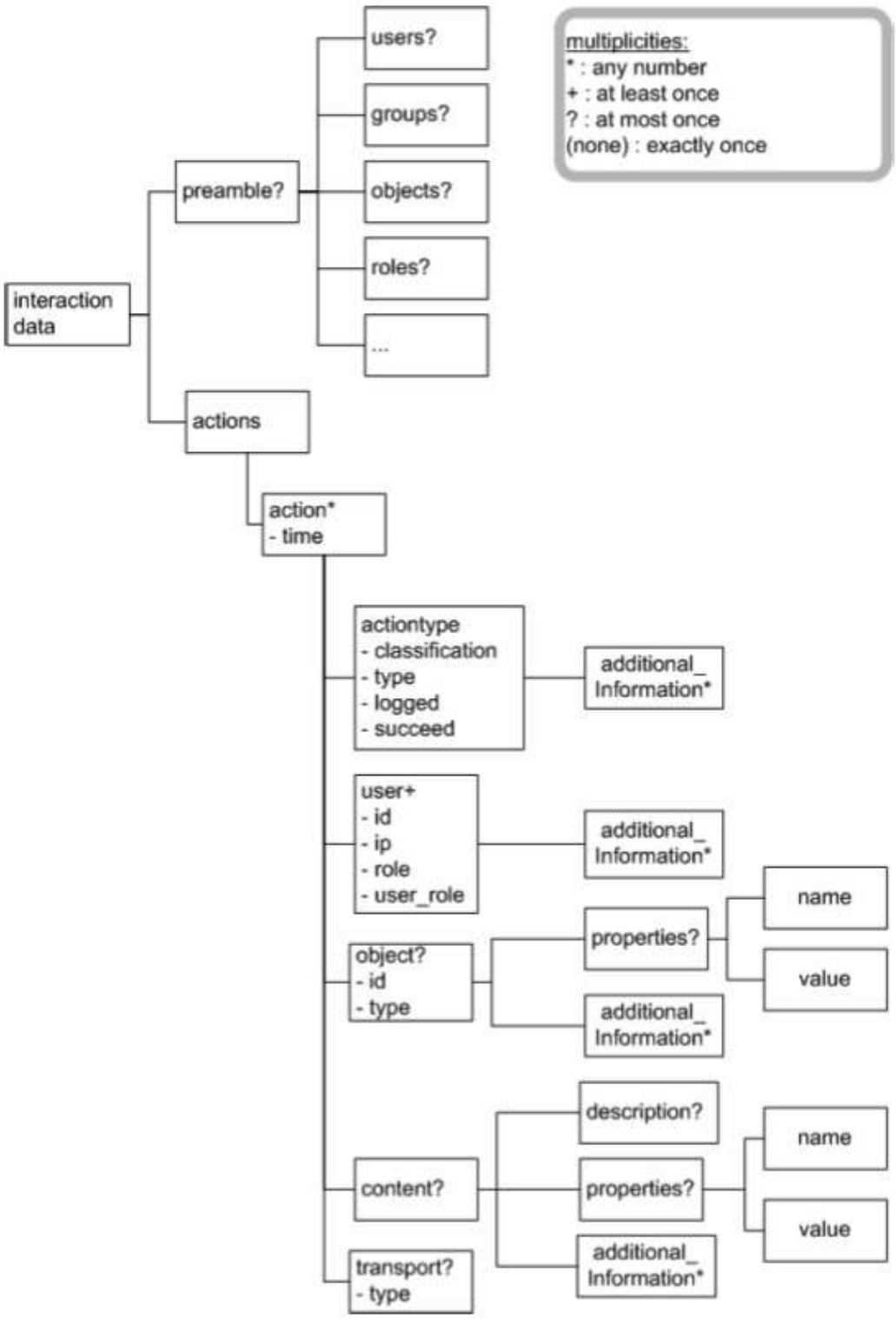


Figure 1-3 Part of the DTD for the Cavicola common format (Martínez *et al.*, 2005, p. 17)

This approach is designed to be used in collaborative computer-mediated situations, both in real-time (i.e. to provide immediate feedback to students and teachers) and in post-hoc analyses. In all cases, the idea is to provide input to a tool which performs some kind of

automated calculation. In order to do so, the tool must have some idea of the semantics of the trace : that it is composed of actions and that these actions have certain properties, such as users, timestamps, etc.

From a perspective of post-hoc analysis, this format has been proposed as a means for interoperability between tools which perform the different steps of an analysis framework proposed by the Cavicola consortium (Harrer et al., 2007; Kahrimanis et al., 2006). In this framework, designed to facilitate discussion of analysis methods and the building of tools to support them, analysis can be broken down into a sequence of steps (cf. Figure 1-4): Capture, segmentation, annotation (and coding), analysis processing (qualitative and quantitative analysis), visualisation and interpretation.

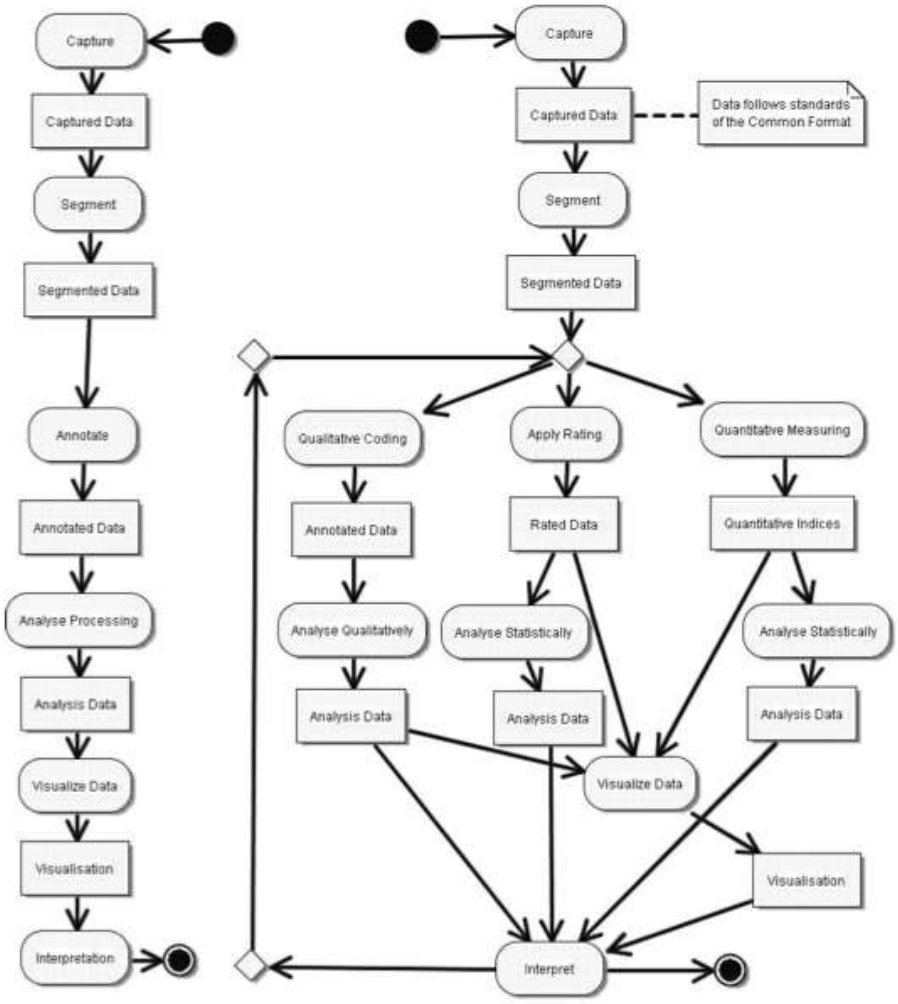


Figure 1-4 The Cavicola analysis process in its generic form (left) and in a more specific instance (right). (Harrer *et al.*, 2007, p.2)

Kahrimanis et al. (2006) note that this model is designed to be flexible to accommodate different kinds of analysis and illustrate this with instantiations of the model in various

situations. For example, in Figure 1-4 , the annotation phase breaks up the analysis into three parallel activities: qualitative coding, rating and quantitative measuring. The qualitative coding is then analysed, allowing the three analyses to be merged back into a single visualisation. The rated data and the quantitative indices are analysed statistically, with all four analysis results (visualisation, qualitative analysis, rating analysis and quantitative analysis) being used to perform the interpretation. They add that in some cases a tool performs multiple steps at once and in others, the same step is performed by several tools.

Discussion. A core assumption of this approach is that some kind of unified semantics of the traces of collaborative actions can be achieved. In order to provide extra flexibility, Martínez et al. (2005) remark:

> *The object description can be enriched as needed by properties, such as the object's attributes and associated values. This provides a very flexible mechanism, but on the backside it lacks the strictness of definition that would be needed to check at a general level if all needed elements for some specific analysis are available. (pp. 15-16)*

In other words, mindful that there will always be some formats that do not completely map onto this common format, they have provided optional elements. But this makes it very difficult for tools which require these optional semantics to explain and enforce their requirements.

Although the Cavicola consortium has reported successful use of this common format on several tools, the format is not currently in widespread usage. The reasons for this are hard to pin down. It could be due to the number of cases requiring special semantics, to the lack of publications regarding the common format (a complete DTD has not yet been published), to the lack of availability of analysis tools which trace-producing tool designers could use (and want to use) to analyse their traces or to a lack of corpora on which analysis tool designers could evaluate and test their tools. The two most likely hypotheses (according to our own experience) are:

- existing trace-producing tools have been developed in such a way that no-one knows the complete extent of the traces they produce, nor how to transform this into the common format, while new trace-producing tool developers have so little available time that they can't invest any into investigating the common format, in spite of the vast potential gains.
- many interaction analysis tools are designed to be used in real-time to provide various forms of feedback to students and teachers. In such a case:

- using a file for communicating between the trace-producing environment and the analysis tool is too slow and resource consuming;
- decoupling the analysis and production tools so hard that creating ad hoc coupled tools is easier;
- the analysis tool usually has to communicate back to the trace-producing environment – and there is no format for doing so.

While we see the value in using the analysis process model to write-up how the analysis was performed after the fact, for example in order to replicate the same analysis on another data set or to report it in a publication, it remains unclear how this model will assist the design of analysis tools. In our own experience, many analyses are much more chaotic in execution, as different methods are explored until a final methodology is stabilised. This has been the experience of others:

> *Our analytical process developed in relation to our research question and though we present it here, it is really part of our findings. Below we describe the main aspects of our analysis, though as it was a process of discovery, the reader should not think that these happened in a linear manner. We moved iteratively back and forth between our research questions and different aspects of data analysis, identifying more and more complexity as we pursued [our topic of interest]. (Kapur & Kinzer, 2009, pp. 26-27)*

1.4.1.2 UTL

Motivation. UTL (Usage Tracking Language) is designed to allow the modelling of "tracks" in a neutral way with regards to educational scenario models, educational environments and log data formats (Choquet & Iksal, 2007a; Choquet & Iksal, 2007b). They define a track as any "datum which provides information on the learning session" and claim that tracks should be modelled in order to be better analysed.

Approach. They propose that tracks should be modelled along three facets: defining, getting and using (cf. Figure 1-5). The *defining* facet allows track users to describe what tracks should be observed and what they mean: their name and purpose. The *getting* facet allows developers to define how such a track is collected or created: does it come from a database, a text file, an XML file, a video file? How can it be extracted from that file? The *using* facet defines how a given track can or is intended to be used: what data fields does it contain? what pedagogical context is it related to? what indicators can be calculated with it? The aim of this distinction is to allow pedagogical scenario designers to define what needs observing

(defining) and for what purpose (using) in order to negotiate with developers on the means of observation (getting).

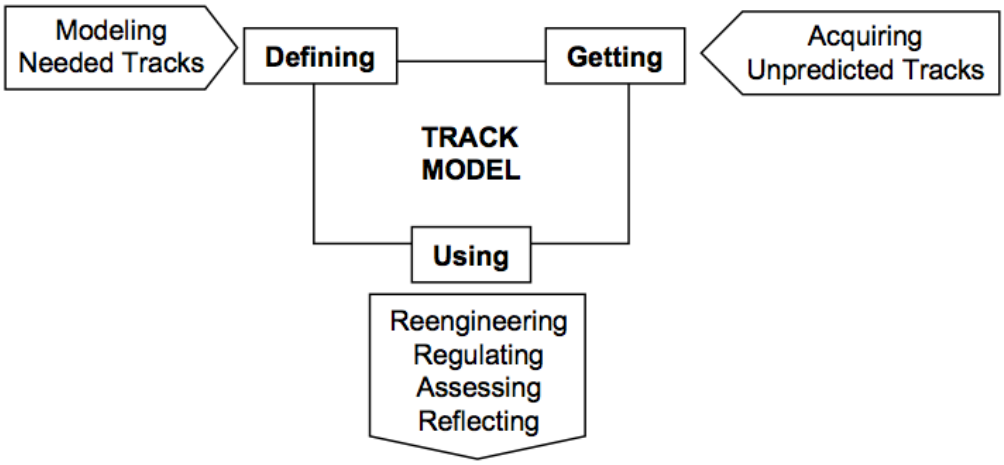


Figure 1-5 The UTL Defining-Getting-Using model (Choquet & Iksal, 2007a, p. 4)

In UTL, various kinds of tracks are modelled in different ways. Choquet & Iksal (2007a) distinguish two kinds of track: *primary-datum* and *derived-datum* (cf. Figure 1-6). A primary-datum can be either a *raw-datum*, found in a log file or database, a *content-datum*, which is some kind of production by a learner, or an *additional-datum*, which is data which is not collected from the learners and which might be useful for the analysis (such as a conceptual ontology). A derived-datum is calculated from some other datum and can be either an intermediate-datum (used as a means to calculate something else) or an indicator: a track defined as having a special meaning within a specific pedagogical scenario or learning situation.

This language can be used not only to model any kind of TEL trace but also to describe the indicators which can be calculated from that trace with a specific analysis purpose in mind. Its authors illustrate its use on several situations, showing how it can be used to deal with the problem of data heterogeneity, to assist in the interpretation of tracks, to help calculate derived data and to compare what the learner did ("descriptive trail") with what the pedagogical scenario designer intended ("prescriptive trail").

Discussion. It is hard to evaluate this work as much of the language still needs implementing. The authors note that they also will need to write authoring and modelling tools for non-technical users. While this work certainly presents a thorough and grounded classification of tracks, it is not clear why this model should be specific to pedagogical situations, or what is to be gained by using this classification in implementation.

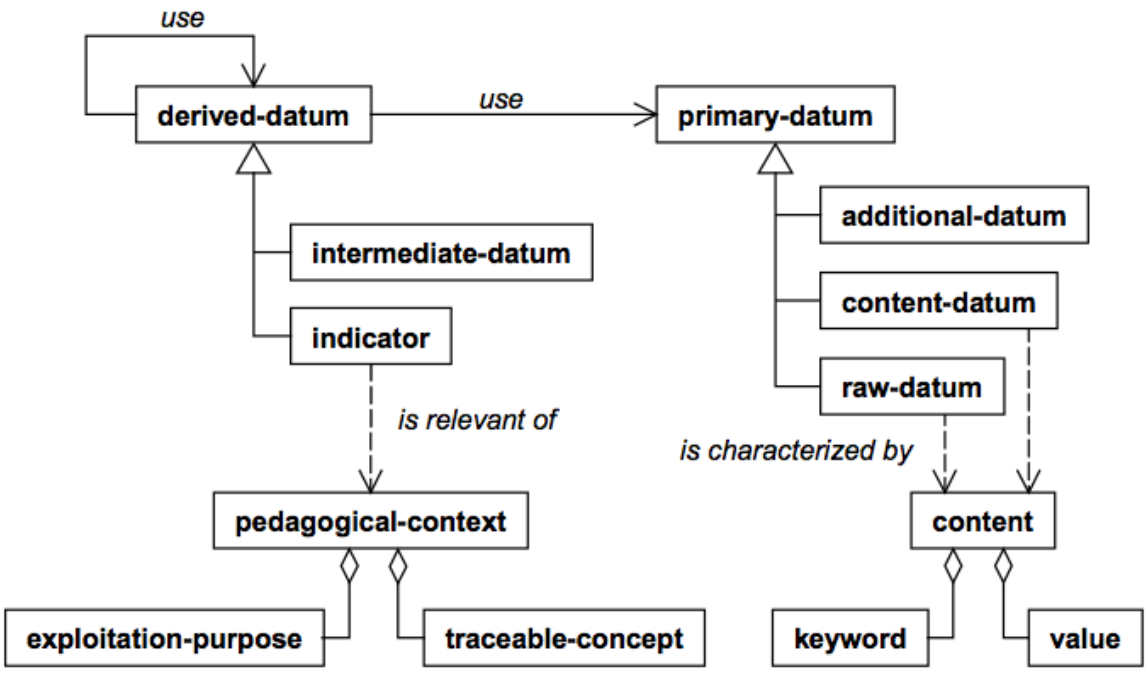


Figure 1-6 The UTL conceptual model of a track (Choquet & Iksal, 2007a, p. 6)

1.4.1.3 MULCE

Motivation. The MULCE project (MUltimodal contextualized Learner Corpus Exchange) stems from the need within the TEL community to share corpora and analyses, in order to enable replication, verification, contradiction and refinement of research results (Reffay, Chanier, Noras, & Betbeder, 2008). The authors specify the requirements on the construction of a LETEC (Learning and Teaching Corpus), by explaining what it needs to contain, how it can be structured and under what conditions it can be shared.

Approach. They first define a corpus, drawing inspiration from linguistic corpora, as everything that is necessary to enable an analysis (cf. Figure 1-7): not only the interaction data (digital traces, video, pre- and post- tests), but also the learning context (current student knowledge, goals, pedagogical scenario, tools used etc.), the research context (research questions, experimental protocol, etc.), the licence under which the corpus is distributed (describing the conditions under which it can be used), and all available analytic work (such as transcription of videos).

Reffay *et al.* (2008) then define an XML structure for defining a corpus, with each component being broken up into three tiers: the description of the component, the reference to the file containing the data regarding that component and the data file itself (not included in the XML file but made available at the URL given by the reference). As far as the data contained in the corpus is concerned, they recommend following existing standards (where they exist), such as

the Text Encoding Initiative (TEI, 2007) for text and dialogue, IMS-LD (IMS-LD, 2003) for pedagogical scenarios, etc. Due to the lack of common formats for digital traces, they propose a format for their own kinds of data, where a session is composed of acts and an act is defined generically, with a different data type for the information specific to each kind of act (email, forum, etc.), leaving other authors to extend it with the acts that can be performed in their own tools (cf. Figure 1-8). The description tier of the corpus should describe as much as possible the format in which the data is encoded.

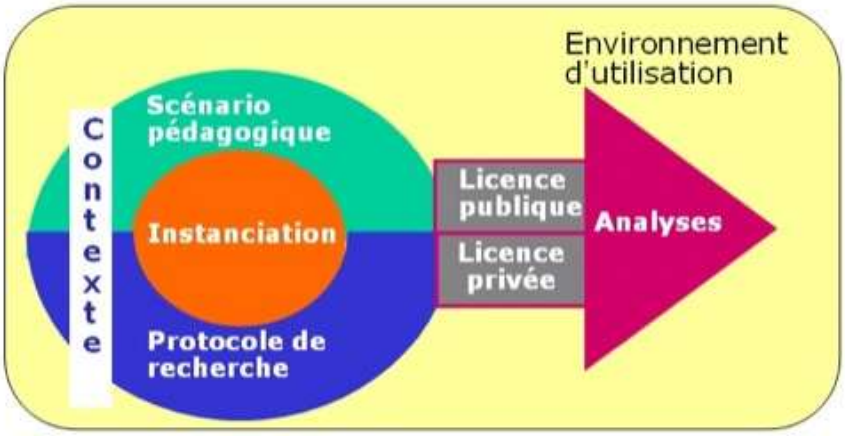


Figure 1-7 Contents of a MULCE corpus (Reffay *et al.*, 2008 p. 6)

As far as the sharing of corpora is concerned Reffay and colleagues highlight various ethical problems. They advise that any shared corps should contain a "private" licence (which defines the rights of the original authors/researchers and states that the corpus is "real" and that the authors are authorised to share it), a "public" licence (defining the rights and obligations of future users of the corpus), an anonymisation policy which describes exactly how the corpus was modified in order to protect the rights of the people being recorded. Furthermore anyone wishing to share a corpus should make sure that they have the appropriate paperwork to show that the people recorded in the study have expressed their consent to being recorded and to that recording being shared.

Tools. This work has led to the creation of an online platform on which corpora and analyses of these corpora can be shared (this platform was on the verge of being opened to the public during the authoring of this dissertation).

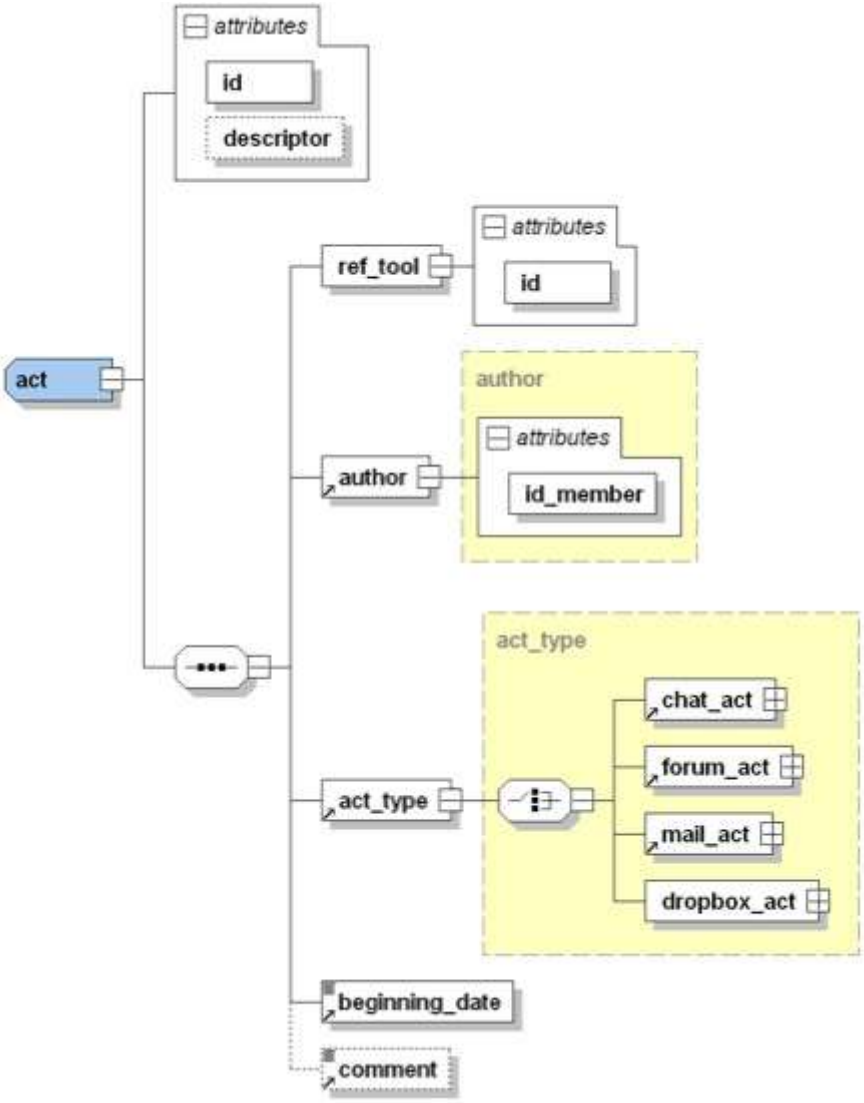


Figure 1-8 Schema for encoding acts in a digital trace in the MULCE project (Reffay *et al.*, 2008 p. 14)

Discussion. This project takes a very pragmatic approach to the modelling of corpora which include traces, contextual information and analyses. For every document there should be a description which explains what it contains, what format it is encoded in and what software can be used to read it. As much as possible, the authors advise the use of standard open formats. While such an approach can only be applauded both for its goals and for the effort to minimise the cost of sharing a corpus (i.e. not necessary to convert it to some common format, or to use a restricted set of tools), by its nature it provides little information on how analyses should be modelled (since there is, as of yet, no standard format).

1.4.1.4 Trace-Based Systems and M-traces

Motivation. This section and the following give an account of work carried out in the LIRIS laboratory in Lyon, in collaboration with many partners on the specification of systems to record, manage and transform digital traces. This work stemmed from the initial double perspective of using trace-based reasoning in a similar way to case-based reasoning (Mille, 2006) and using traces in order to personalise the experience of a learner in a TEL situation (Settouti, Prié, Mille, & Marty, 2006). Case-base reasoning (Schank, 1982) is a technique for solving new problems by drawing on experience in already solved problems ("what did we do in such a case?"). Trace-based reasoning is the idea that, as traces store past experience, they can be leveraged to solve new problems, benefiting not only from the automatic storage of experience but of that experience being stored in a temporally situated way ("what steps did we follow in the past when we were faced with such a problem?").

Approach. Champin, Prié, & Mille (2004) initially proposed a framework for modelling traces called *Musette* (Modelling USEs and Tasks for Tracing Experience) in which a trace is composed of a sequence of *states* and *transitions*, each transition being composed of one or more *events,* and each state composed of one or more *objects*. The effect of a transition is described by the relationship between events and objects. The model of the trace describes the kinds of events, objects and relationships that can exist within a trace and affords the transformation from a trace into an *explained trace*. *Explained task signatures* (common patterns which make some kind of sense) can then be defined from the model and found within the trace. In more recent work (Settouti *et al.*, 2006), a trace (or what we are calling a *digital trace*) is defined as a *temporal sequence of observed elements*, meaning that these elements are events which occur and are recorded in a temporal sequence. This lays the foundation for a *trace-based system*, designed to record, store, transform, query and visualise traces (cf. Figure 1-9).

In this architecture, a collection model defines the means of recording a trace into the system (much like inserts into a database). A trace carries with it a model which describes the semantics of its contents, thus defining the transformations that can be applied to it (with a transformation model), the queries that can be made on it (with a query model) and the visualisations that can be made (with a visualisation model).

Settouti, Prié, Champin, Marty, & Mille (2009) extend this work, formally defining the notion of a trace model, a trace, a query language and a transformation language. They define a *trace*

model as a vocabulary for representing traces: how time is represented, the categories observed events can belong to and the attributes they can have, the relationships that can exist between events, and the kinds of events they can link. A trace which conforms to a model (or *M-trace*) is then defined as a set of events and relationships, each event having a unique id, an associated time and a set of named attributes; the events and relationships are typed according to the definitions in the model.

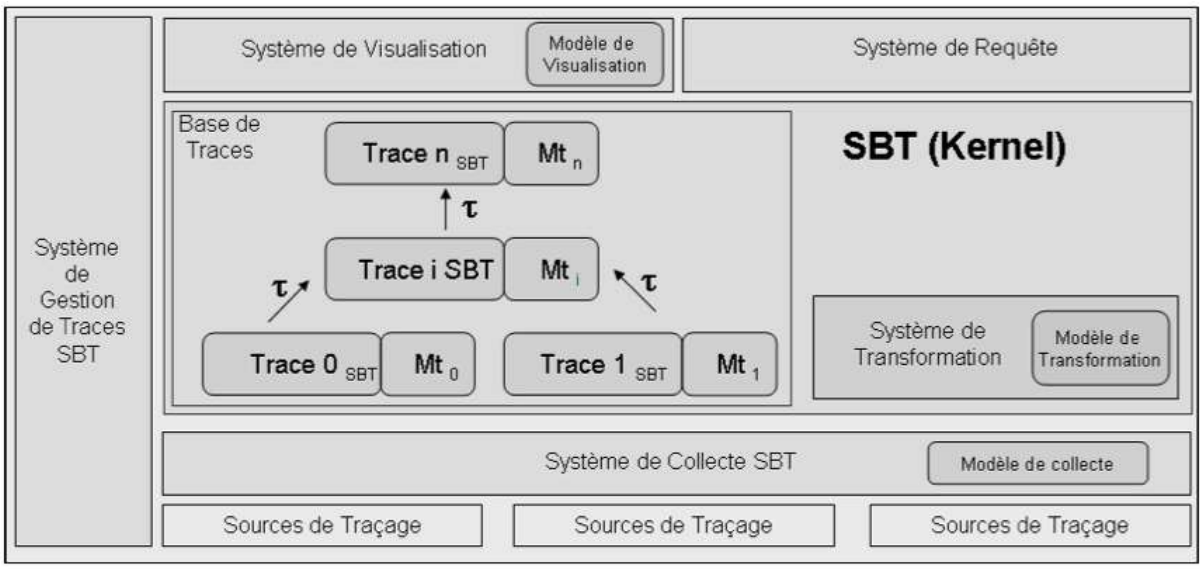


Figure 1-9 Architecture of a trace-based system (Settouti *et al.*, 2006, p. 5)

Settouti and colleagues define a grammar for describing patterns : informally this is a tuple of constrained variables (bound variables, free variables, and variables involved in relationships with elements from a trace or trace model, or other variables). The execution of that pattern returns all the tuples of variables that match the constraints. A query is defined as a pattern which contains no events from a particular trace (thus rendering it generic over any trace conforming to the model used in the query), a name and a subset of the variables which must be returned. A transformation is defined as an operation taking one or more M-traces (each conforming to a given model) and producing a new M-trace (conforming to a new model) by applying a set of transformation rules. A transformation rule is composed of a query and a template for producing new trace elements from the result of the query.

As this system is designed to be used for all applications using traces (being in spirit rather like a database management system for traces), Settouti and colleagues take care to differentiate *off-line* and *online* M-traces. An offline trace is considered to be the final result, meaning that queries can be run once. An online trace arrives in "real time" in bursts of new elements. A continuous query is defined as a query which runs on an online trace – and

needing to be re-evaluated as new elements arrive. Clearly, running a query on the complete dataset each time would be costly – in many cases, it is possible and more efficient to partially compute the results on the existing dataset and increment the result set based on a calculation which is applied only to the newly arrived elements. This leads to the examination of the monotonicity of queries : queries whose existing results will not be invalidated by newly arrived elements are monotonic; queries such as "A not followed by B" may return a result that is valid at a given moment (A has happened) and be later invalidated (B suddenly happens). In other words, it is impossible (or at the least misleading) to evaluate non-monotonic queries (e.g. identify participants who have not used a particular tool) on incomplete data as such results may later be invalidated. Furthermore it becomes difficult to re-use the results of previous computations when new elements arrive.

Finally, Settouti and colleagues show how to map the formal semantics they have defined onto the Datalog language (A query and rule language for deductive databases; Gelfond & Lifschitz, 1991), thus providing a means of implementing patterns, queries and transformations.

Discussion. This approach is particularly interesting as it examines why traces need to be studied more closely in order to define a generic means of defining calculations on them (necessity of a model, difficulty of applying transformations in presence of a model, problems of evaluation on online traces). It has been successfully applied to certain kinds of traces, in order to define specific transformations on a given trace model and to assist the transformation of traces from one model to another (Djouad, 2008). However, it has not, as of yet, yielded a generic tool for use by the community. While the approach has been developed in a TEL context, it can be applied to traces from any domain. The context seems to be more due to the first need for such a system being felt within the TEL community than to any close relationship between TEL and the modelling of traces.

1.4.1.5 ABSTRACT

Motivation. A major contribution to the work on M-traces and trace-based systems described in the previous section is the ABSTRACT (Analysis of Behaviour and Situation for menTal Representation Assessment and Cognitive acTivity modelling) system and the models behind it (Georgeon, 2008; Georgeon, Henning, Bellet, & Mille, 2007). Georgeon studied car driving from the perspective of cognitive ergonomics, more specifically in the discovery of patterns in motorway driving and overtaking. Georgeon worked on a very complex data set including the

state of the car (speed, direction, brake-pressure, indicator usage, proximity to other vehicles), gaze data from the driver (looking straight ahead, in the rear view mirror, in the wing mirrors, in the blind spot, etc.) and video data. In order to discover these patterns in his recorded data, he found it necessary to build a system which would allow both the exploration of the data and the definition of transformations, abstracting from the raw data up to the patterns of interest.

Approach. Georgeon's approach to analysis of traces stems from the double perspective of ESDA (exploratory sequential data analysis – of which we give an overview in §1.4.5.1) and knowledge management/engineering. Georgeon cites Bachimont (2004) to explain that knowledge engineering should rather be described as a means to engineer the *inscription* of knowledge. The role of knowledge engineering is to design systems which, rather than automatically recording, transforming or producing knowledge, give its users the means to do so via symbolic representations of knowledge. Georgeon further draws on the concept of Knowledge Discovery from DataBase (Fayyad, 1996): "the overall process of discovering potentially useful and previously unknown information or knowledge from a database". The cycle of knowledge discovery (cf. Figure 1-10) is piloted by the user, who is the only person qualified to say whether and why a statistically significant pattern in the data represents new knowledge.

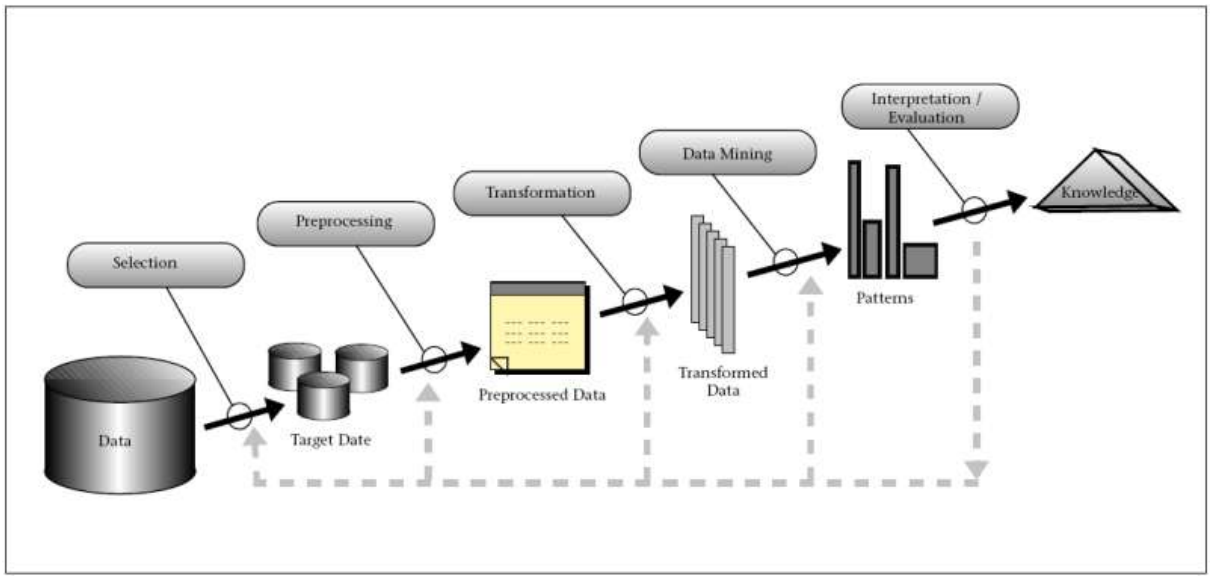


Figure 1-10 The Knowledge Discovery from DataBase cycle (Fayyad, 1996, p. 41)

Georgeon explains that his goal is to "set up a system allowing the discovery of knowledge from traces of activity" and insists on the central role of the user of the system as an agent in

this process and on the necessity of modelling the traces of activity and the new actions according to an ontology.

In the ABSTRACT system, traces of activity are modelled as an ordered graph which is explained by an ontology (i.e. each vertex and arc can be attached to a concept in an ontology). In this graph, vertices are objects recorded in the data or created during the analysis, and edges are relationships between these objects (again, these relationships can come from the recording or have been created during the analysis).

Tools. Georgeon combines a number of existing technologies to put this model into operation. The trace is represented as an RDF (Resource Description Format) graph written in an XML (eXtensible Markup Language) format and modelled by an RDF-Schema ontology. The SPARQL (SPARQL Protocol and RDF Query Language) query language is used to define patterns and construct new elements in the trace. The trace is transformed via XSLT (eXtensible Stylesheet Language: Transformation) into an SVG (Scalable Vector Graphics) image. Finally, ontologies are constructed and edited by the user via the Protégé tool.

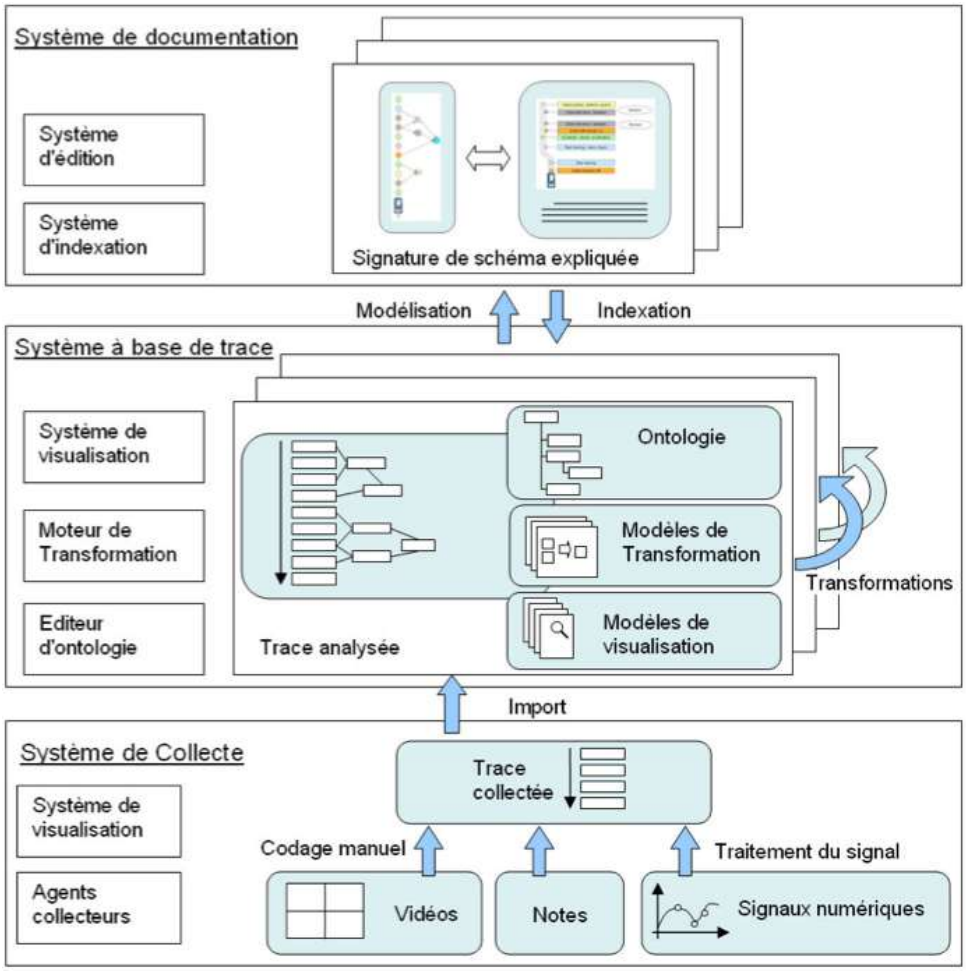


Figure 1-11 The ABSTRACT architecture (Georgeon, 2008, p. 140)

These technologies are combined in an architecture containing three components (cf. Figure 1-11): a trace collecting system, designed to combine the recorded data into a single, modelled trace, a trace-based system, designed to allow the transformation, modelling and visualisation of traces, and a documentation system for explaining the transformations developed by the user and the explained trace signatures that have been discovered.

The visualisations which are created in this system represent different kinds of objects as different-shaped and -coloured symbols on a graphical timeline. The abstractions which are produced as a result of transformations from lower level elements are linked to these elements, allowing easy navigation up and down the construction of abstractions. In Figure 1-12, the bottom half shows a large time period while the top half shows a 10 second slice of that period. The circles at the bottom of the top visualisation indicate the initial trace elements from which the other more abstract elements, represented by triangles and squares, are calculated. This visualisation can be synchronised with the video and new elements can be added by defining new transformation rules.

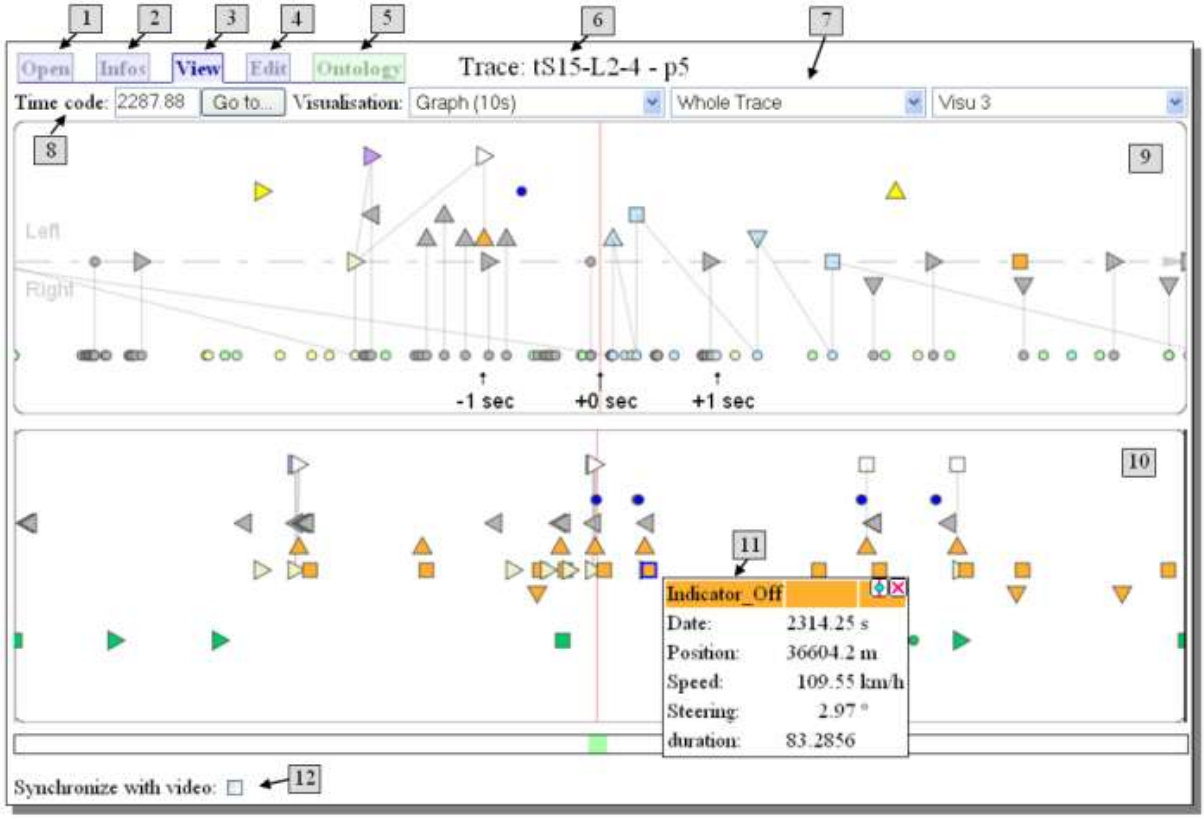


Figure 1-12 Visualisation in ABSTRACT (http://liris.cnrs.fr/abstract)

Georgeon explains that the knowledge that can be created in ABSTRACT is implicitly contained in the source trace and that the analyst's role is to make that information explicit

through increasing levels of abstraction. He notes that while this tool was used to analyse driving in an ergonomics context, it was designed to be used for any kind of ergonomics study and could probably also be used for analyses in other fields.

Discussion. This approach is, to our knowledge, the first complete traced-based system to have existed and is one of the few cases of such a system actually being completely implemented and used in real analysis situations. Georgeon notes that usability is critically important for such a system to be used by ergonomists. He highlights the key issues of ABSTRACT as being : the necessity of some abstraction to be built on top of SPARQL to avoid users from having to know how to use it; the limits of the technologies in use, rendering some calculations very slow; and the current lack of integration between the different modules (visualisation, definition of transformations, definition of the ontologies, etc.).

1.4.2 Automatic analysis techniques

In this section we examine techniques from various fields in computer science which have successfully been applied to analysing traces, particularly those of TEL data.

1.4.2.1 (Educational) data mining and machine learning

Data mining is a term applied to various techniques used to automatically discover patterns in large quantities of data. Mining of web usage logs is a natural expansion of the mining which is performed on data from retailers in order to discover buying patterns (e.g. people who buy cars often buy petrol). This has lead to the development of sequential data mining techniques (Masseglia, Teisseire, & Poncelet, 2005), which look for common patterns across the activity log data of many users.

Another recent development is that of an educational data mining community (Educational Data Mining, n.d.), attracting members from the intelligent tutoring community in particular (intelligent tutors are software specialised in providing customized instruction or feedback to assist a student perform a particular learning-related task (Psotka, Massey, & Mutter, 1988)). Mostow & Beck (2006) explain that an intelligent tutoring system such as their own (Mostow & Aist, 2001) had logged over 50 000 sessions of interactions between learners and the system. They explain how to record educational data in order to make it suitable to being mined.

- Data should be logged at multiple grain sizes as data at different granularities is suited to answering different questions.
- As much data as possible should be *reified* (translated into something machine-understandable): for example replacing handwritten input with typed input, freehand drawing with drawing from a palette.
- Timing can provide extra information : not only does it show when something happens and in which order, but also, for example, how long it took for a student to answer a particular question.
- Analysis of students' writing can be used to grade answers to questions.
- In many systems, students discuss with their peers. If this information is recorded it can be used to quantify frequency, volume and patterns of communication.
- Information about students such as age, gender, IQ, prior knowledge, etc. can all benefit the automatic discovery of correlations.
- While it is not always feasible, manual labelling of certain sessions can help establish heuristics for automated labelling, or be used to discover correlations with other variables.
- Various probes can be added to access information that is otherwise not directly observable. For example, reading comprehension has been successfully measured by deleting words to turn sentences into fill-in-the-blank questions.
- Once systems have been deployed at a large scale, it is possible to isolate the success factors in learning outcomes by introducing randomized trials: during a given session (or part of a session), a randomized decision is made (such as giving or not giving a hint). Over a large number of trials, the outcomes can be compared.

Mostow and Beck continue to explain three ways of transforming the recorded data into mineable episodes. The event stream can be segmented into shorter episodes that can be analyzed individually, for example by breaking up the session after each student response. Slicing, rather than reducing the length of episodes, reduces their breadth, by examining a limited number of variables at a time. For example, if multiple competencies are evaluated, one "slice" can be created for each skill. Finally, each occurrence of a local decision defines an experimental trial. A data stream can be broken up into experimental trials, provided each includes the context (who, what, when), the decision and the outcome. For example, each of 10 questions of a quiz filled out by a student could become a trial, associating the student's

name, past reading and activity and the question itself with the outcome – whether the question was answered correctly or not.

Mostow and Beck then present four techniques for mining the episodes that have been created. Human browsing of interactions can help identify phenomena of interest which can lead to the generation of hypotheses which can then be tested. Aggregating data with respect to some feature (e.g. time spent thinking about the answer to questions compared to time spent reading) can lead to correlations with learning outcomes. A model can be made to fit a data set by finding the parameter values which minimize prediction errors. Finally, a model can be trained on a given dataset in order to predict learning outcomes. Mostow & Beck (2006) report a number of situations where their techniques have successfully been used.

Mostow, Beck, Cuneo, Gouvea, & Heiner (2005) have created a tool called the Session Browser, which considers that logged events can best be understood by placing them in hierarchy which describes the context. Events can be searched for and are then displayed as children of their ancestor events – A being an ancestor of B if B happens during A.

In a similar vein, several machine learning techniques have been applied to existing datasets in order to train a model which predicts learning outcomes or learning styles. Nüssli, Jermann, Sangin, & Dillenbourg (2009) have shown that signal-level data such as the congruence of eye-tracking data for two collaborators working in different locations and their speech recordings can be used to predict performance in certain collaborative situations. McLaren *et al.* (2007) have experimented with the automated analysis of discussion graphs created by students in order to provide information to tutors and teachers about the *kinds* of discussions which are happening.

One of the most frequently used analysis methods is to manually categorise turns in computer-mediated discussion according to a given coding scheme (this can be considered a form of reification, usually used to prepare for statistical analysis). This categorisation is very time consuming. Rosé *et al.* (2008) and Erkens & Jansen (2008) have independently reported success in using certain linguistic analysis techniques to extract features from discussions which can then be used to perform automated coding.

Harrer, Hever, & Ziebarth (2007) provide a tool for the automated and human-assisted discovery of recurring patterns in a trace file, while Romero, Gutiérrez, Freire, & Ventura (2008) and Reimann, Frerejean, & Thompson (2009) use process mining techniques to discover and describe patterns respectively in web browsing and group decision-making

processes. Reimann (2009) explains in depth how such techniques can be used to make sense of time and sequence in traces.

In spite of the increasing availability of such automated analysis tools, they are not yet seeing widespread use, perhaps because of technical difficulties in using them, or because the results they give are difficult to interpret. They do however show an enormous potential which should be exploited in designing support for analysis.

1.4.2.2 Social Network Analysis

Social Network Analysis (SNA) uses graph analysis techniques on a graph where vertices represent users and edges represent relationships between users (such as "sends a message to", "reads a message from", "becomes friends with") in social software (Wassermann & Faust, 1994). These analytic methods have in turn been borrowed from quantitative sociologists.

In the CSCL community, these techniques have been used to measure the cohesion of collaborative learning groups (Reffay & Chanier, 2003), to qualify communication structures and correlate them with final outcomes (Harrer, Zeini, & Pinkwart, 2006), to visualise the difference between groups and to detect roles, such as that of leader, coordinator, etc. (Martínez, Dimitriadis, Rubia, Gómez, & de la Fuente, 2003).

In order for SNA methods to work, it is important to define what actions should be considered as relevant in order for relationships between users to be created.

1.4.2.3 Knowledge Space Visualiser (KSV)

Knowledge Space Visualiser (Teplovs, 2008) is a tool built upon the Knowledge Forum (Scardamalia, 2003) to explore the data in the forum in different ways. Knowledge Forum is a forum which exists in a two-dimensional structure; in other words, messages are not only linked to their parents but have a position in a two dimensional space (chosen by the author or rearranged by other members of the forum). This information is stored in a tuplebase (Teplovs, Green, & Scardamalia, 2008): a database in which each row is a tuple of named attributes – rows do not necessarily contain the same set of attributes (or columns) and applications which do not know about certain attributes are free to ignore them. In a first step to prepare the data for visualisation, Teplovs applies latent semantic analysis (Landauer, Foltz, & Laham, 1998) techniques to establish the semantic distance between messages. As a

result of this calculation, new relationships are inserted into the Knowledge Forum tuplebase, describing the semantic distance between each pair of messages. This supplements the existing parent-child and message-author relationships which already exist in the tuplebase.

The tuplebase is then visualised using the KSV software which projects the messages into a two dimensional visualisation using a force-directed layout. The user can specify which relationships should be considered important for the layout, enabling the creation of semantic clusters (cf. Figure 1-13), clusters by user, or a "prettification" of the Knowledge Forum space (i.e. taking the original knowledge forum two-dimensional layout and graph structure and "smoothing" it out using a force-directed layout).

As other Knowledge Forum modules can add new attributes (for example categorisations of nodes) to existing forum messages in the tuplebase, they can also be visualised in KSV. Furthermore, KSV is not limited to Knowledge Forum, but can read and visualise any XML data with enough similarity to forum data.

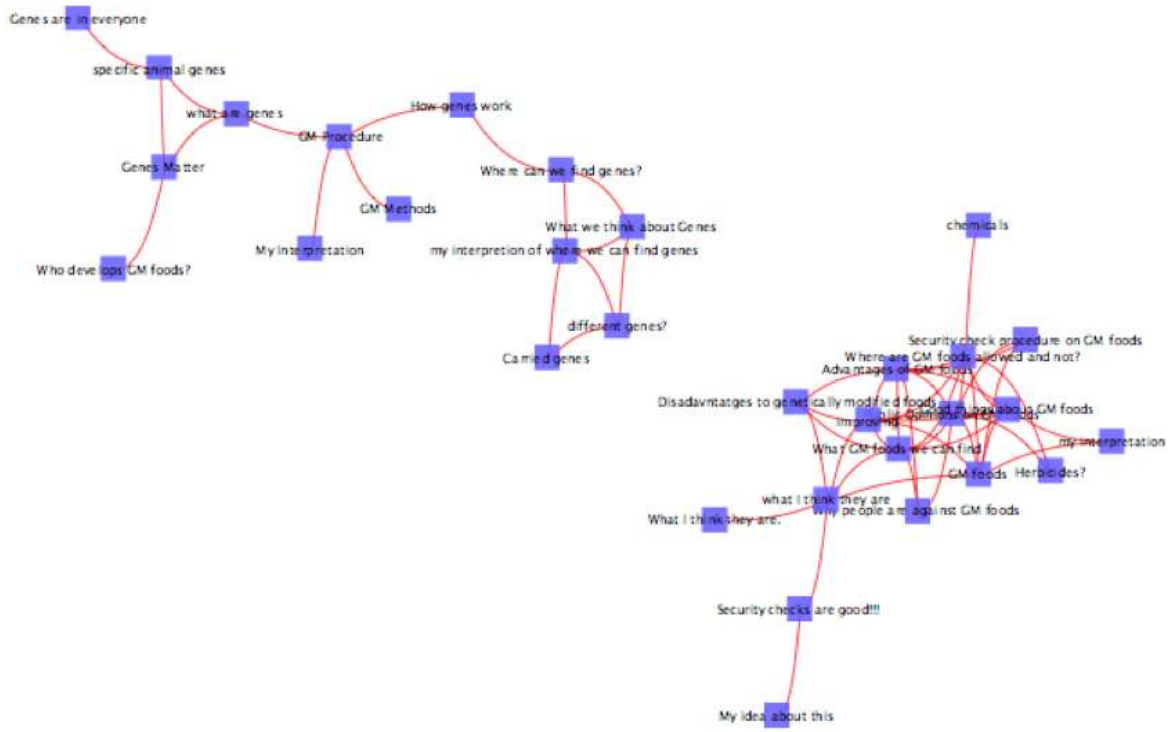


Figure 1-13 Semantic Clusters in KSV, where red arcs indicate a semantic proximity which is higher than a certain threshold and blue boxes represent messages (Teplovs, 2008, p. 7).

1.4.3 Audio-video analysis

Many fields linked to the study of human behaviour base their analyses on audio and video data: linguistics, cognitive science, ethnography, didactics, etc. As such, many researcher-created and commercial tools exist for the analysis of audio and video data. Aside from signal-level treatment of audio and video, which we will not address here, such tools are mostly concerned with the transcription (of speech, gesture, facial expressions, etc.) and annotation (linguistic features, types of acts, etc.) of audio and video sources. An extensive survey of such tools is presented by Dybkjaer *et al*. (2001) and more recent work includes tools such as Elan (Wittenburg, Brugman, Russel, Klassmann, & Sloetjes, 2006) and the Nite XML Toolkit (presented below). Dybkjaer and colleagues recommend that a tool to support the annotation and transcription of *Natural Interactivity and Multimodality* (NIMM) data fulfil at least the following specifications:

- Flexible and open architecture;
- Separation of the user interface from the internal application logic;
- Transcription support;
- Annotation support at different levels;
- Powerful functions for query, statistical analysis and information extraction;
- Synchronised visualisation;
- Easy to use interface;
- Support for addition of new coding schemes and for defining new visualisations;
- Possibility of importing existing data sources.

Reidsma, Jovanovic, & Hofs (2005) explain that different analysis needs will lead to the need for different tools. These different analysis needs explain the main differences in functionalities between the tools reviewed by Dybkjaer and colleagues. Annotation layers can correspond to features of two natures: direct observations which might potentially be codes in real-time and interpretations which require more reflection. Annotations can refer to explicit features (which are clearly identified in some other layer like speech acts, gestures, etc.) or to implicit features (which are present, for example in a video and thus only identified by timestamp and filename). Where possible, explicit features should be directly linked to their annotations. The characteristics of segmentation (in particular of implicit input layers) define whether segmentations can overlap whether they must be continuous, whether they must

cover the whole of the input layer. Annotation types can be more or less complex, more or less numerous. This will define to what extent it is possible to map different annotation types to hotkeys. Tools can define relations between annotated elements, such as adjacency pairs, questions and answers etc. These can be difficult to show in a user interface and can be used to construct powerful queries. Finally, constraints can be imposed to enforce certain kinds of integrity on annotation sequences and relationships.

Reidsma and colleagues explain that they do not survey statistical analysis functionalities as these are covered by existing software packages, nor import/export, coding scheme structures and queries as they belong to a separate class of problem and are solved by other tools.

One such tool, the NITE XML Toolkit or NXT (Carletta *et al.*, 2003), is a set of libraries and tools designed to provide for the native representation, manipulation, query and analysis of complex annotations on multimodal data. A typical example of such annotations can be seen in Figure 1-14.

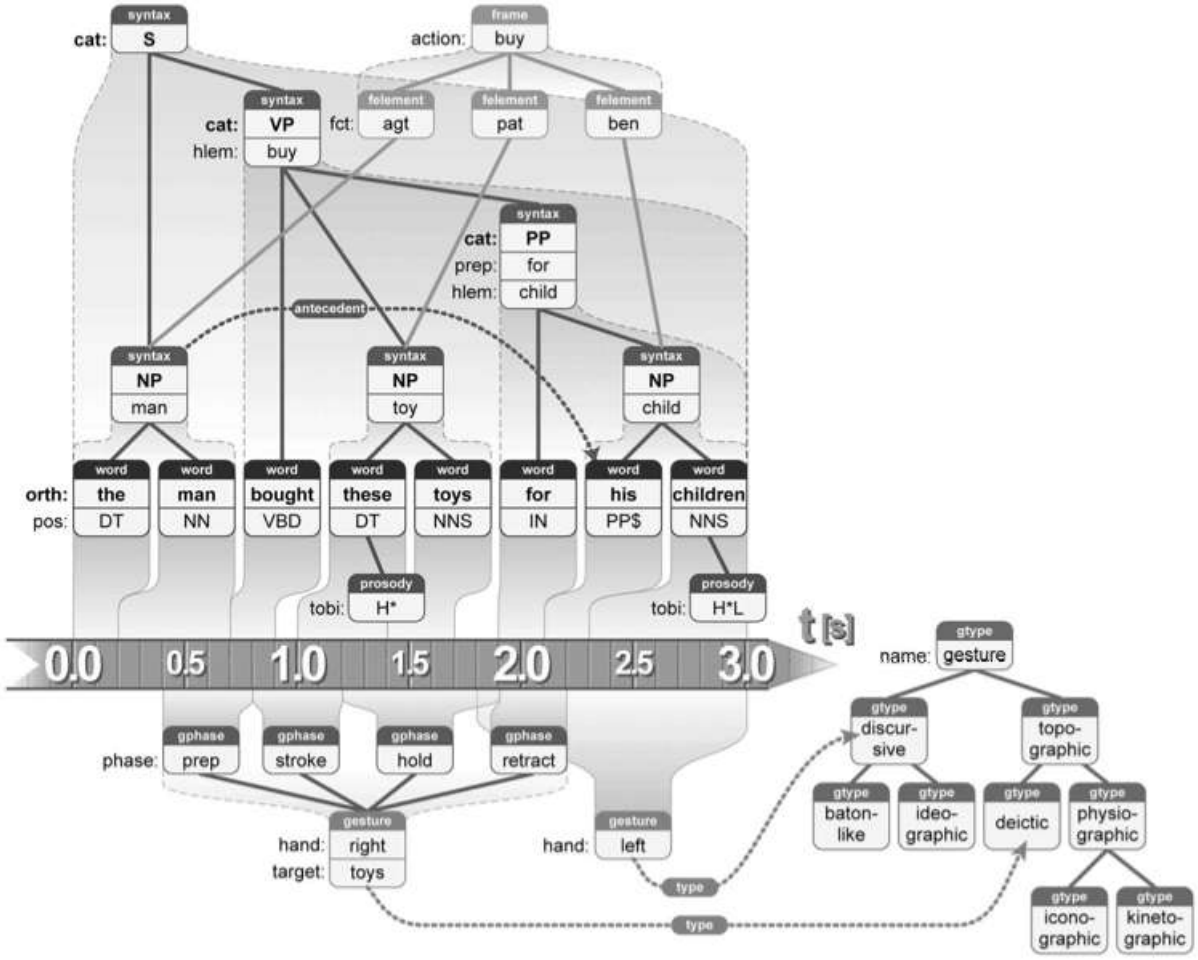


Figure 1-14 Sample annotation trees described with the Nite XML Toolkit model (Carletta *et al.*, 2003, p. 354)

In this example, an audio signal is annotated for both gesture and speech. The bottom level of each of these *annotation trees* is linked to the audio signal. The next levels are related in a tree structure, for example, grouping together a personal pronoun (his) and a noun (children) into a nominal phrase. Annotations can be linked together, for example in order to make explicit the antecedent of "his", to serve as a "source signal" for a new annotation tree (enabling cross-annotation of the same data), or to create a type system (an annotation being linked to a new tree which represents the type) (Carletta, Kilgour, O'Donnell, Evert, & Voormann, 2003).

The data structure (implemented in one XML file for each *annotation tree*) considers a number of annotation "objects", with each object having a type, a set of key-value pairs, a set of relationships to other annotation objects (child-parent or other, more complex relationships) and an optional time interval (possibly inherited from child annotations). Only the bottom "layer" of an annotation tree can link to elements from other annotation trees. A metadata file describes the constraints of each tree (what key-value pairs, relationships, etc. can exist).

The NXT also describes a query language, specially designed to perform queries on NXT data. A basic query describes a tuple of variables and rules on those variables. The execution of a query returns the set of tuples for which the rules evaluate to true. Many tools have been designed which use NXT as a library for storing and querying and visualising annotations on specific types of data, such as gesture, part of speech tagging, etc.

The tools described in this section were all designed with audio-video data in mind and, as such, are not necessarily directly applicable to numeric traces. The kind of analyses they are designed to assist are most often linguistic in nature, and tend to be principally focused around relating a source signal to (often complex) annotation structures. These tools are an example of a comprehensive effort to and create software to support such a particular analytic activity and, as such, can be a major source of inspiration for our work. Aside from the model proposed by the Nite XML Toolkit, however, the discussion in the litterature is limited to comparing and recommending sets of features. This is largely disappointing as it does not inform us on how these features should be integrated into a coherent analytic framework.

1.4.4 Trace analysis tools

In the previous section we looked at tools specifically designed for audio and video analysis (and with no facilities for digital trace analysis). Before looking at tools designed to analyse both kinds of data, we will examine tools designed only for the analysis of digital traces. These tools assist the analysis of "distant" interaction, often in asynchronous situations (forum discussion for example) or for analysing data which has already been transcribed from the video.

1.4.4.1 CORDTRA

Hmelo-Silver, Chernobilsky, & Jordan (2008) explain that in CSCL data, learning is often distributed among participants, media and modalities. As such, simple coding and counting of discourse acts (oral or computer-mediated) does not give a full account of how learning occurs. They propose the use of CORDTRA diagrams (Chronologically-Ordered Representations of Discourse and Tool-Related Activity) to explore data sets by juxtaposing a variety of activity codes on a single timeline (cf. Figure 1-15). Each row represents a chronologically ordered series of representations of events in a given media or modality. Such diagrams are intended to enable humans to explore the data which is represented and discover patterns or significant events in that data.

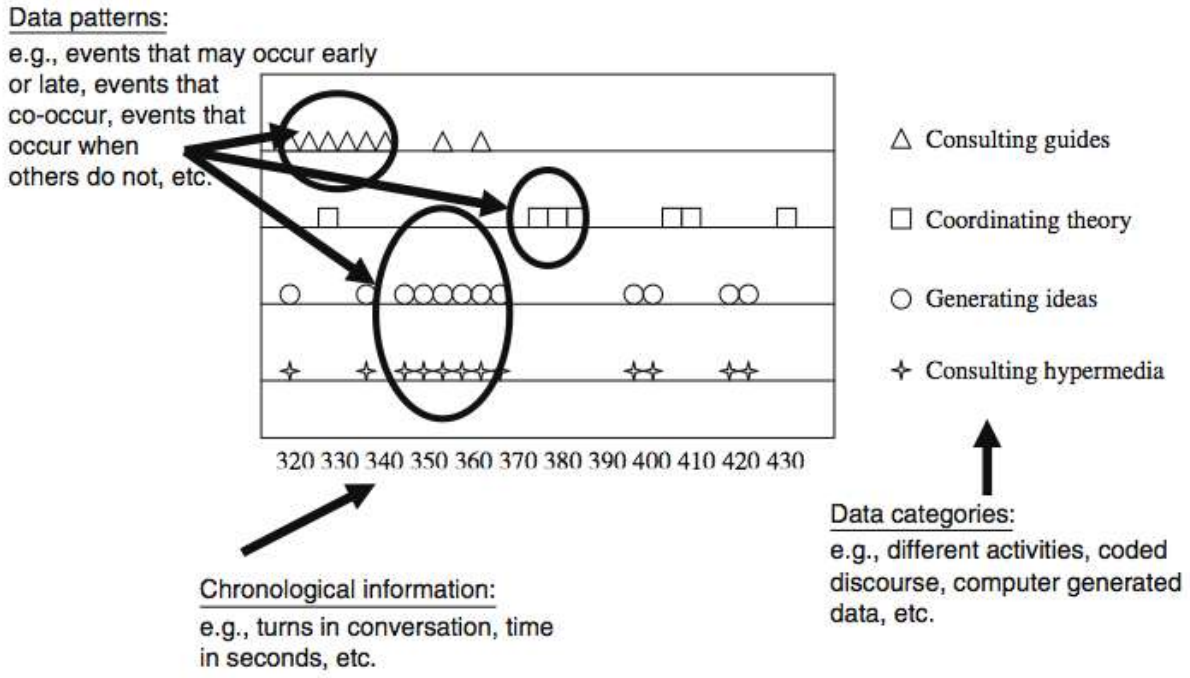


Figure 1-15 Explanation of CORDTRA diagrams (Hmelo-Silver *et al.*, 2008, p. 412)

CORDTRA diagrams are generated from an appropriately structured Excel file. Hmelo-Silver and colleagues note the effort needed in preparing the data for diagram generation and the difficulty in finding patterns in such diagrams.

1.4.4.2 Forum Analysis

Forums are used to such an extent in educational settings that there exist many tools dedicated to their visualisation, exploration and analysis.

May, George, & Prévôt (2008) propose an architecture for tracking, analysing and visualising online communication (TrAVis), in forums in particular. They explain that, in order to analyse such data, it is important to have information on how a message is read and written (time spent, whether the whole message was read or not, etc.). This enables them to generate visualisations which give detailed information about the reading activity of a user (cf. Figure 1-16), allowing the user of this platform (a researcher or a tutor) to better understand whether each message was read in detail or whether some of them were not read all the way through.

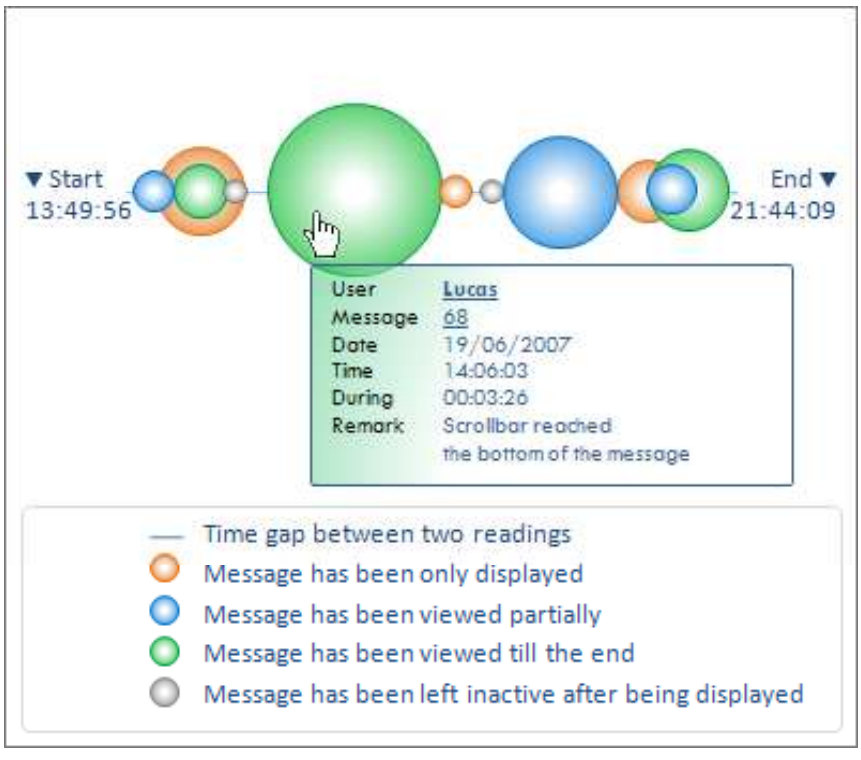


Figure 1-16 Visualisation of message reading with TrAVis (May *et al.*, 2008, p. 184)

This architecture is generally applicable to computer-mediated communications (part of the work of May and colleagues has been to propose a general model for computer-mediated

communication) and has been put into place for the tracking of forums and their subsequent analysis and visualisation by learners and tutors.

ViCoDiLi (Teutsch, Bangou, and Dejean-Thircuir, 2008) is a tool for visualising and exploring online discussion corpora. It allows its users to better understand messages by placing them in the context in which they were produced (i.e. in a given discussion thread), and also by giving a common interface to explore communication through different media. It provides a number of means to assist qualitative analysis: letting users *decontextualise* messages or short exchanges by placing them with other messages in a structure which makes analytic sense; providing references to corpus extracts which can be shared in a meaningful way; formatting a text to highlight what an analysis would like to emphasize. ViCoDiLi also provides anonymisation and search procedures.

Calico (Giguet, Lucas, Blondel, & Bruillard, 2009) is a website for the sharing and analysis of forum data. It proposes a common format for forum data, including an import module which ensures that the participants are anonymised (as per the recommendations of the MULCE project). Once data is in Calico, it can be visualised in a generic way, either following the posting order or the reply structure. Various analysis modules enable visualisations, keyword search, statistics and role discovery of users.

These tools exploit the fact that forums are so widely used that it is possible and useful to establish a common generic format for such data. In order to use this approach for more generic traces, it would be necessary to extend the model to include other kinds of computer-mediations. As the Cavicola consortium has shown (cf. §1.4.1.1), this is a challenging task.

1.4.4.3 Tool replayers

As we have already noted, when analysing computer-mediation there is a choice to be made between using digital traces and screen capture. While both would ideally be used, screen capture can be costly in terms of experimental setup (setting up the recording apparatus and then centralising the recordings) and storage space. Digital traces, on the other hand, even when all necessary interactions are traced, can be difficult to understand. One solution to this problem is to use a *tool replayer*. A replayer can read the trace produced by a given tool and re-create what the user saw on screen. In the case of computer-mediated communication, such tools are easy to produce as it is enough to create an "observer" mode of the trace-producing tool in which messages can only be read (Corbel *et al*, 2003). This will not be exactly what the original user saw (unless mouse-movement, etc. is also traced and reproduced) but should

be enough to give the analyst sufficient insight into the meaning of a given event in the digital trace and how it fits into the wider context. As Zemel and Cakir (2009) note, referring to the replayer available for analysing the Virtual Math Teams environment (Stahl, 2009), a replayer ca sometimes be indispensable in understanding precisely how actions unfolded.

The advantages of tool replayers extend beyond the smaller file size of digital traces. In a distributed context, a single trace can be enough to understand what happened on all users' computers. Furthermore, tool replayers can be interactive, allowing the user of the replayer to resize the window, use scrollbars, and visit various tabbed panes in order to not be limited to seeing what one particular user saw. This affords significant potential for comparing the activity of several users. For example, if scrolling were traced, a replayer could show several "shadow" scrollbars representing the current view of each user (cf. Figure 1-17). Finally, because replayers reproduce the internal state of the tool, they can also access information which was not shown to the initial users or present that information in a different manner. One such possibility is exploited in the DREW (Corbel *et al*, 2003) shared text editor, in which newly contributed text is briefly displayed in the colour of the user of the environment (for awareness purposes) before fading to black. In the replayer, however, this text keeps its colour, thus making it easier for the user of the replayer to remember who wrote what. In the DREW argumentative graph editor, many objects (e.g. boxes with text in them) are being constantly created, edited, moved and deleted. In order to "help" the analyst, the trace will refer to an object with a unique id (e.g. box 27.1). But which object is box 27.1 in the replayer? In order to help the researcher, the ids of objects are displayed when replaying.

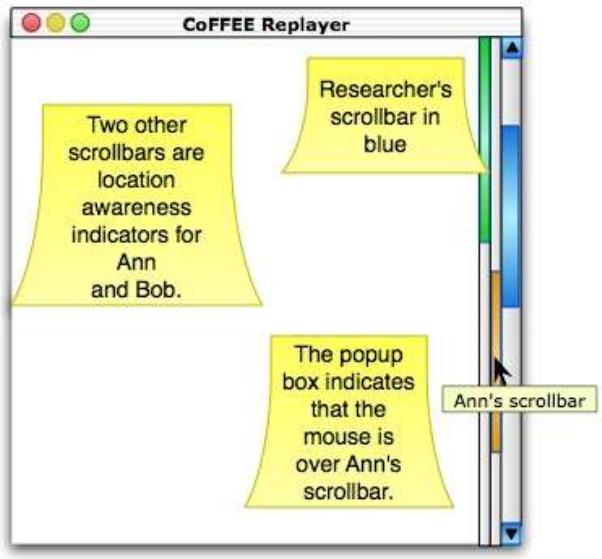


Figure 1-17 Mock-up of a replayer interface allowing the researcher to see what two users saw at a particular moment.

Replayers can be seen as an useful, easy to implement, means of showing a variety of information to researchers. They benefit from the knowledge embodied in the tool about the relationship between its activity and the produced trace and afford a means, not only to bridge the gap between trace and understanding, but also to provide additional information which would not be present in a screen capture video.

1.4.5 Exploratory sequential data analysis and replay tools

In this section, we will examine tools and methods which combine audio-video and digital trace analysis. These tools are mostly used in the field of human-computer interaction (both research and industrial uses) and in various kinds of "e-social science" which explore the uses of new technology, both as a collaboration/communication means and as a means to record new kinds of data (eye-tracking, lie-detection, heart-rate, GPS position).

1.4.5.1 Exploratory Sequential Data Analysis

Exploratory Sequential Data Analysis (ESDA) is a term coined by Sanderson & Fischer (1994) to describe a variety of techniques for the analysis of sequential data (where the chronology of the data is meaningful) in a fashion which is exploratory – at least initially. ESDA follows the general process described in Figure 1-18. In an external loop, the scientific process consists of asking questions, collecting and analysing data, all within an epistemological framework which constrains the nature of questions which can be asked, the data to be collected and the acceptability of the generated statements. An inner loop, for data analysis, concerns the iterative creation of products which have been transformed from the recorded data (or from products created in a previous iteration). In this inner loop, analysts attempt to *"[smooth] the data – manipulating it so that its essential structure becomes apparent"* (Fischer & Sanderson, 1996, p. 28).

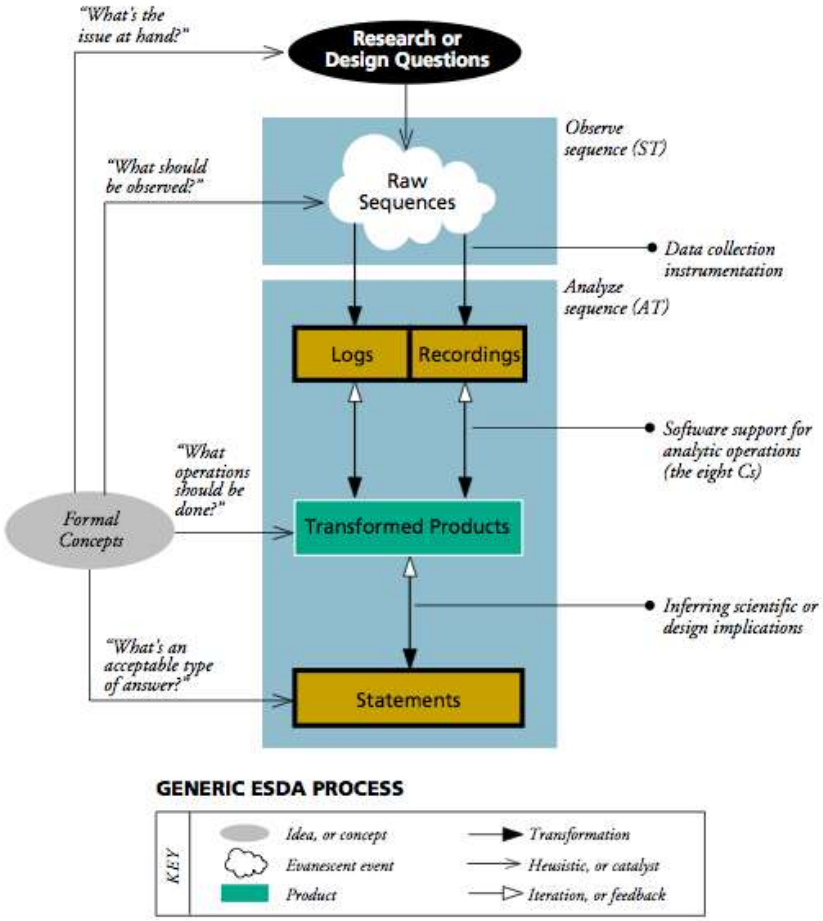


Figure 1-18 The general ESDA process follows an outer loop linking research questions to statements and an inner analytic loop in which data is iteratively transformed to allow the generation of statements (Fischer & Sanderson, 1996, p. 26).

Sanderson and Fischer describe eight "smoothing" operations which they call the "eight Cs" (cf. Figure 1-19): chunks, comments, codes, connections, comparisons, constraints, conversions and computations. *Chunks* concern the segmentation and re-segmentation of data, potentially in a hierarchical structure. *Comments* are unstructured, informal annotations, which are attached to data, to chunks or to other intermediary products. *Codes* are a way of abstracting the data, capturing its meaning while reducing the variability and lack of precision of its vocabulary. *Connections* describe relationships between the data such as temporality, the implicit or explicit structure of the data (e.g. the implicit reply structures in a chat and the explicit reply structures in a threaded forum), or the relationship between different types of media (e.g. video and digital traces). *Comparisons* examine the data through differentiating lenses such as the same coding by different analysts, data from different experimental conditions or between a predictive behaviour model and the described behaviour. The *constraints* applied to the data can be used to filter it so as to only show, for example, a selected interval, a particular medium or data coded with a certain keyword. *Conversions*

transform data in order to present it in a different way – a different coding scheme, a different graphical representation, a different grain of analysis. Finally, *computations* reduce the data to summary representations: statistics, average values, etc.

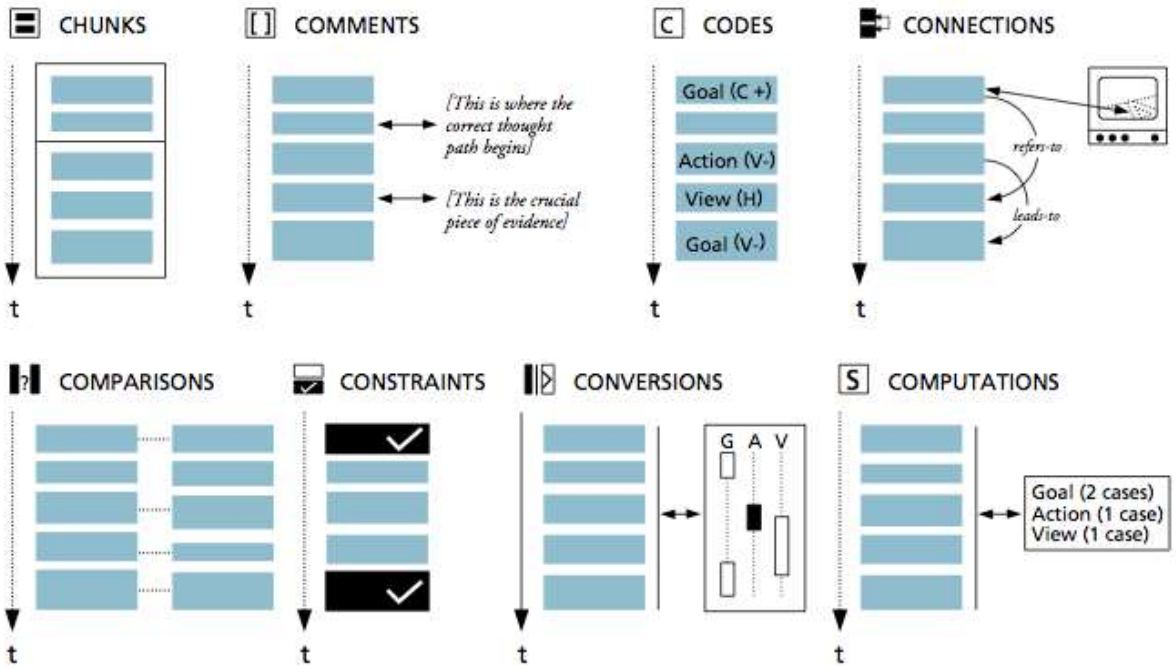


Figure 1-19 The eight Cs – eight operations for transforming data to reveal its essential structure. (Fischer & Sanderson, 1996, p. 29)

Fischer & Sanderson (1996) note:

> *"By putting these fundamental operations together in different ways, according to need, rather than by slavishly following a particular technique, the analyst has a good chance of crafting a new methodological approach that will meet his or her needs." (p. 30)*

They add that the richer analyses often make use of all these operations. The principal reported difficulty of ESDA is what Sanderson and Fischer call the AT:ST (analysis time: sequence time) ratio. They note that many analysts are discouraged at the prospect of spending between five and one hundred times the recording time to perform their analysis.

They present some of the reasons for such a large analysis time and explain some possible ways of resolving these problems. Often, unnecessary steps are performed such as detailed transcription which will not necessarily be exploited. They suggest that with appropriate navigational aids, it could be enough for analysts to listen to audio data or transcribe only where it is later necessary. Another culprit is making premature commitments to analyse large quantities of data in a given way (e.g. a particular coding scheme) which actually turns out to be uninformative. Sanderson and Fischer suggest that a thorough analysis should first be

carried out on a small subset of the data in order to determine whether it is both feasible and useful to perform that analysis on all the data. They note that because of our increasing ability to collect large amounts of data, we often feel that it should be completely analysed. They suggest that thorough analysis of part of the data may be more appropriate than a more limited analysis of a complete dataset. The final issue highlighted by Sanderson and Fischer is that of poor software support. They note that software often constrains the kinds of analyses that can be done, either by providing better support for it or through lack of support for alternatives. For example, when only coding is available, annotating (using an ad-hoc category for each annotation) becomes difficult; if it is not possible to create connections, analyses which explore the non-linear structures in the data will rarely be used.

Other issues which are reported as limiting the effectiveness of ESDA include: insufficient domain knowledge by the analyst, lack of knowledge of available ESDA techniques and data which is not rich enough to contain the information which is essential to provide evidence for a given statement.

1.4.5.2 MacShapa and other HCI analysis tools

Based on the needs and operations they defined for ESDA, Sanderson *et al.* (1994) built MacSHAPA. In MacSHAPA, a data stream is organised as a series of time-aligned variables in a spreadsheet (cf. Figure 1-20). Each column corresponds to a different variable and each row to an interval in time. Rows in different columns can have different heights. From a data stream, it is possible to create various forms of reports such as statistics (counts and aggregate durations of certain event types), pattern analysis (lag sequential analysis, transition reports) or comparisons (e.g. between two coding schemes). Queries and filters can select certain rows in the data or search for certain patterns and create new data streams from the results. Data can be edited in the spreadsheet, imported from external files and then visualised (e.g. in a graphical timeline). Finally, spreadsheet and graphical timeline visualisation can be synchronised with each other and with video.

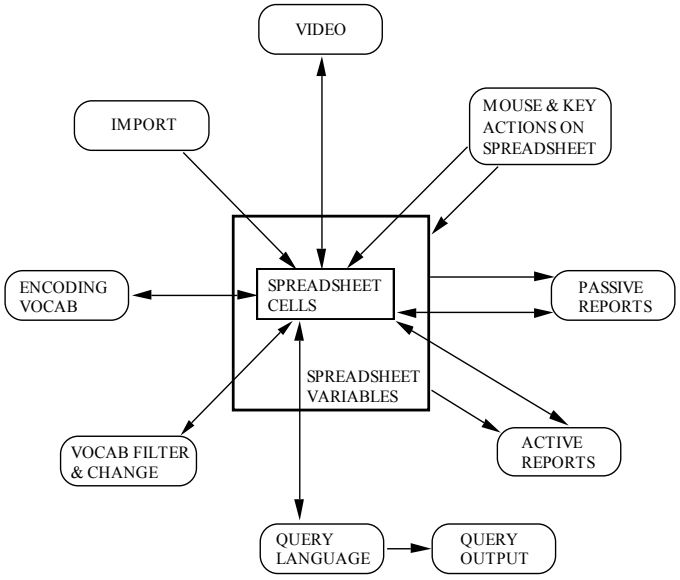


Figure 1-20 Summary of functionalities in MacSHAPA (Sanderson, 1994, p. 5)

This tool saw significant use during the 1990s, particularly in usability community. A multiplatform equivalent called OpenSHAPA has been under development for several years but is not yet available.

Other tools of similar nature (e.g. Badre, Guzdial, Hudson, & Santos, 1995) are reviewed in Hilbert & Redmiles (2000). They classify the techniques featured in various tools into: synch and search, transformation, counts and summary statistics, sequence detection, sequence comparison, and sequence characterisation, visualisation, and integration. The strengths and limitations of various techniques are examined. They conclude that while many tools offer some of these techniques, none offer all of them and, in particular, there is a marked lack of support for automated transformation.

In synch and search, events in different media are synchronised, with searches in one medium being used to locate information in other media. The limitations lie with the necessity of human intervention, particularly in the case of video and in the (obvious) necessity of the media under study to contain the correct granularity or abstraction of information.

Hilbert and Redmiles identify three kinds of transformation: selection, abstraction and recoding. Selection is the restriction of a data stream to certain kinds of events, abstraction is the creation of new events at a higher level of granularity, based on lower level events, and recoding is the process of defining new event streams upon which further operations can be done from the results of selection or abstraction. They note that such operations are essential in order to see meaningful patterns emerge from the "noise" of the source data, but warn that

such operations can also throw away data which might have been meaningful. They also deplore the lack of support for automation in the processes of selection and abstraction.

Counts and summary statistics abstract away the temporal nature of the data and allow certain immediately obvious results to be shown. They are, however, not very good at dealing with the sequential nature of the data. Hilbert and Redmiles also note that while many tools incorporate certain kinds of counts and statistics, few of them allow ways to define new ones.

Hilbert and Redmiles define sequences or patterns as being either concrete (a sequence defined by the user of the tool as a list of values) or abstract (linked to a model, such as a state-transition diagram). They distinguish the operations of sequence detection (finding elements in a data stream which match a given sequence), sequence comparison (calculating the proximity of a sequence to a reference sequence) and sequence characterisation (finding common sequences in a data stream and modelling them as an abstract sequence). Sequence detection and comparison are limited by the fact that sequence definers must already know which sequences to look for. Sequence comparison is complicated by the variety of metrics which can be used to compare sequences. Sequence characterisation is problematic in that many sequences which are detected by automated means are difficult to make sense of from an analytic perspective.

Visualisations are ways of presenting the data which make use of humans' innate visual analysis abilities to interpret the data. The number of possible visualisations of any given dataset make it unlikely that a tool will provide the visualisation means adapted to particular needs, making it necessary for many visualisations to be built by hand.

Finally, the integration between all these elements is crucial for an analysis to be carried out without obstacles and to mutualise the benefits of different techniques. Of all the surveyed tools, MacSHAPA goes the furthest in integrating its capabilities together, but even so, lacks features for automated support of transformation, abstraction and recoding.

While the features examined in this section provide advanced means for digital trace analysis combined, to a certain extent, with video, they are designed for cases of human-computer interaction, rather than interaction between humans, possibly mediated by computers. These are notably different in that human computer interaction logs often have events with few properties (time, event-type) selected among a fairly large controlled vocabulary (mouse moved, mouse clicked, key pressed, menu selected, etc.), while computer mediated interaction events rather have many properties (time, event-type, sender, group, receiver,

contents, etc.) with either a small vocabulary or a large uncontrolled vocabulary (such as sentences).

1.4.5.3 SAW

Developed by Goodman *et al*. (2006), the Synchronized Analysis Workspace (SAW) is designed for the exploration of multiple synchronised data streams (audio, video, transcripts, screen capture and digital traces). It provides both graphical timeline and list visualisations of events. These visualisations can be restricted by filters and augmented by a supplementary data stream within which annotations are placed. SAW is unique among the tools of this state of the art in that it explicitly provides for collaborative analysis through shared filters and annotation streams. SAW also provides means for exporting data, images and workspaces for subsequent analysis by other tools.

1.4.5.4 ActivityLens

ActivityLens is a tool which stems from the necessity of combining digital traces and video for the analysis of many kinds of collaborative learning scenarios (Avouris *et al.*, 2007; Fiotakis *et al.*, 2007). It allows the synchronised viewing of digital traces and video, providing an interface to describe how the timestamps of different media should be aligned. Different means for manipulating data include the addition of comments, filters to restrict the view of digital traces to certain actors or tools, and a way of re-constructing two levels of abstraction on top of the digital trace (cf. Figure 1-21).

The three levels of observation of activity are drawn from activity theory (Bertelsen & Bodker, 2003). Digital trace data and additional comments are placed at an *operations* level (defined as responding to changes in environmental conditions). A number of operations-level events can constitute an *action*, which is performed with a specific goal in mind. In a similar manner, actions can be combined into *activities*, which describe the activity from a strategic or motivational viewpoint.

While ActivityLens has successfully been used for the analysis of many cases of collaborative activity and learning, it is strongly biased towards analysis from a multi-level viewpoint, such as activity theory. This three-tiered structure has also been subverted to other uses (e.g. Rummel, Meier, Spada, Kahrimanis, & Avouris, in press), but lacks the flexibility needed to support different kinds of analysis (notably coding and automated transformation).

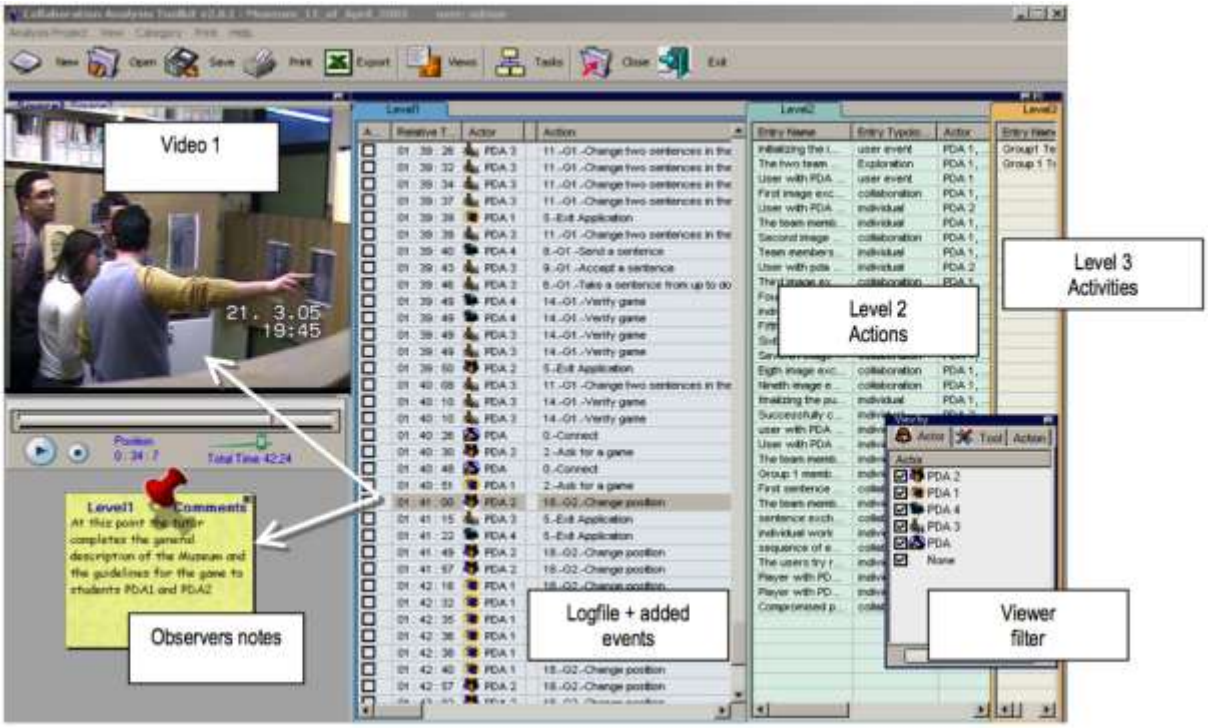


Figure 1-21 The ActivityLens environment showing a multi-level view of the situation under analysis. (Avouris *et al.*, 2007, p. 18)

1.4.5.5 Replayer and DRS

Finally, we present two tools which focus on the synchronised replay of multiple views of data.

The first, Replayer (Morrisson *et al.*, 2006) has been used in particular to study the use of collaborative mobile applications. It provides several visualisation components whose views are synchronised (potentially across multiple computers) in a *brush and link* fashion (Becker & Cleveland, 1987): selecting an element in one view will highlight matching elements in other views. These components include media bridges to video viewing software on various operating systems, time series to plot numerical data by time, event series to view various events on a graphical timeline, histograms to show data from a non time based perspective, and a bridge with Google Earth, allowing events to be plotted on a map (cf. Figure 1-22).

DRS (Digital Replay Software) extends the notion of multiple synchronised replay to the inclusion of user-created annotation and transcription of the data (Greenhalgh, French, Humble, & Tennent, 2007). In DRS, an analysis project comprises a set of data resources (video, audio, log, etc.) and transcriptions and annotations of that data. All the data in a project can be replayed synchronously (Figure 1-23). DRS also provides search features to look for codes and words in transcriptions and annotations. A set of annotations on a given

media is considered a form of media in its own right, with each annotation being defined as a content (plain text or code) attached to a region of a media file, defined by a begin and end time within that media.

Figure 1-22 In this example from Replayer, the events selected in the left hand view are also highlighted on the map and in the video timeline. (http://ww.dcs.gla.ac.uk/~morrisaj/Replayer.html)

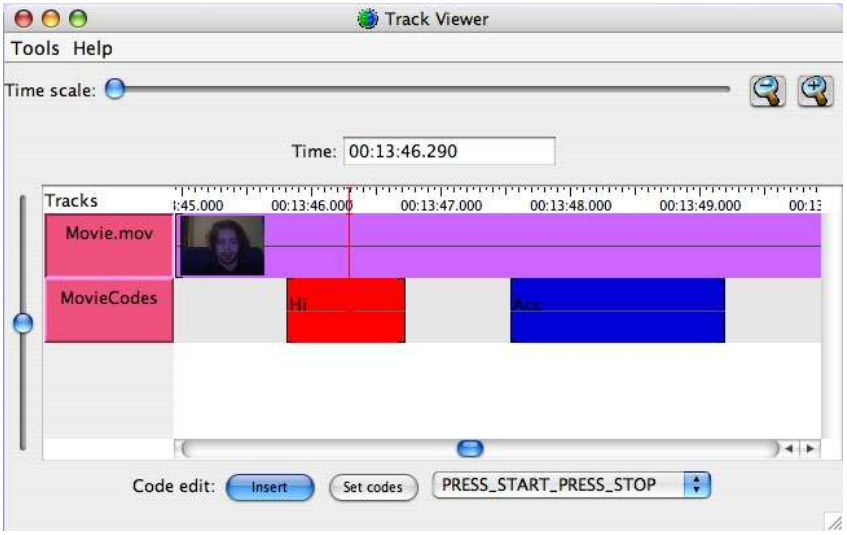


Figure 1-23 In DRS, most kinds of data can be viewed in the track viewer. Here the transcription and the colour codes of the coding of this transcription are synchronised with the video. **(http://ww.mrl.nott.ac.uk/research/projects/dress/software/DRS/Home.html)**

DRS also provides import and export features, including a *log file workbench* which allows various forms of log-file to be easily converted into a DRS database. Visualisation of data in DRS can be in the track viewer, in a time series view (for sensor data such as heart rate) or in a spreadsheet view for transcription and annotations.

Both these tools can be applied to many kinds of time-based data and are specifically designed for replay and manual annotation, but with little automated support for analysis.

1.5 Discussion of the state of the art

In this section, we examine the state of the art in relationship to the question of capitalising on analyses of CSCL data, in order to highlight both the most important, reusable elements, and the shortcomings we have revealed. Our work, presented in this dissertation, draws inspiration from these reusable elements and addresses some of the shortcomings. We will centre this discussion around three major themes: what we know about analysis in CSCL, the models for representing traces and analyses, and what we can learn from existing tools.

1.5.1 What we know about analysis in CSCL

In the state of the art, analysis has been presented in a variety of ways. Knowledge Discovery from DataBase and Exploratory Sequential Data Analysis give two highly generic explanations of what constitutes analysis within a certain context. The Cavicola process model describes a certain class of CSCL analyses while Educational Data Mining explains how to prepare educational data in order to apply data mining techniques. Audio-video analysis tools exist within a context where analysis has already been defined as the creation of complex annotations on the recorded media. Finally, various tools and analysis methods contribute their feature sets to a list of operations which can be performed during analysis.

What emerges quite clearly is that analysis is an iterative process wherein the analyst creates increasingly abstract representations of the data which eliminate the "noise" in the original data and reify the analysts' knowledge both of the domain in general and of the situation under analysis in particular. This reification of knowledge can be done in several ways, such as annotation, abstract groupings of lower-level events or through decontextualising elements of the data and re-uniting them thematically.

Replay tools such as SAW, the DREW replayer, ActivityLens, DRS and Replayer, along with visualisation tools such as ABSTRACT, Listen, KSV, social network analysis, TrAVis and ViCoDiLi illustrate the necessity of presenting data in ways which researchers can understand in order to further analyse and interpret it.

What is less clear, however, is in what way CSCL data is "special", whether support for analysis in CSCL is different from support for analysis in other fields and, if so, how it differs

and how tools should support this difference. The Cavicola process model suggests that analysis can be described as a set of sequential operations, which is at odds with what ESDA and KDDB tell us about analysis. Georgeon (2008) also insists on the necessity of flexibility and agency by the analyst:

> *Parmi une possibilité infinie de filtrages, encodages, transformations des enregistrements d'observation, c'est l'ergonome qui choisit d'explorer ceux qui paraissent pertinents par rapport à ses objectifs finaux. [Among an infinite possibility of filterings, codings and transformations of the observational recordings, it is the ergonomist who must choose to explore those which appear pertinent with regard to his final objectives.]* (p. 33)

On the other hand, while ABSTRACT and MacSHAPA are tools which match their respective models of analysis, they are intended for their own specific fields of analysis (ergonomics and usability respectively) and there is no certainty that this carries over to CSCL. Mostow & Beck (2006) explain how to prepare educational data for mining. However, they do not clearly explain in what way "educational" data is special, apart from each trial being associated with a *learning outcome* which measures the learner's progress, knowledge or success and attempts to correlate it with other factors. ActivityLens is specially designed for CSCL contexts, but conforms to a very specific pattern of analytic knowledge building, inspired by activity theory. Finally, TrAVis and ViCoDiLi are designed for specific forms of computer-mediated learning (specifically computer-mediated communications). We have not found any tools or models which provide an account of analysis which is claimed to apply particularly to the CSCL context without it being further restrictive to some specific subset of CSCL (a form of computer mediation or of analysis).

The other shortcoming of this information about analysis is that the more generic the model for analysis, the fewer hints it gives about how this model can be operationalised. In particular, while the eight Cs of ESDA are compellingly presented to cover a wide variety of analytic needs, and could certainly cover many kinds of CSCL analysis, they are difficult to implement directly. The fact that MacSHAPA's functionalities are not mapped directly onto the eight Cs is a first indication of this. Further evidence is given by the fact that some of these operations need to be covered by multiple functionalities, while some functionalities can be used for more than one operation. For example, structural connections (such as the reply structure in a discussion) can be created by relational annotation (either manual or automated) and must be visualised in some way to become apparent; furthermore, temporal connections must often be shown through synchronisation, although they can also be created through automated transformation (merging data streams while preserving temporal ordering) and

subsequently visualised. Conversely, queries can be used for conversion (e.g. convert from one data format into another), comparison (e.g. comparing coding by two different users) and computation (e.g. counting the number of times each user intervenes). Hilbert & Redmiles (2000) show one example of how these operations can be summarised by functionalities of synch and search, transformation, counts and summary statistics, various sequence related functionalities and visualisation. But the final functionality they describe is "integration support", leaving open the question of how these functionalities can be implemented in an integrated way.

1.5.2 Models for representing traces and analyses

Among the models for representing traces, we have seen the Cavicola common format, the Usage Tracking Language and trace-based systems. Models for representing analyses include ABSTRACT's use of trace-based systems, UTL's means of representing interaction indicators and the NXT model of annotations. We also examined MULCE, a model for representing learning corpora.

Trace-based systems seem to overcome the limitations of the common format (difficult to extend) and address the modelling of traces in a more elegant way than UTL. UTL takes a more pragmatic approach, including in the trace model information on the format in which it is stored and in the uses it is intended for. Trace-based systems, however, make claims of generality with regard to manipulating traces, but do not claim to know which manipulations are of use for a given analysis situation. Aside from the representation of corpora in MULCE, however, non-digital traces are not included in these models.

ABSTRACT's modelling of analyses in a trace-based system suggest that all analyses can be performed through iterative transformation of the trace (and be represented by a trace). This may be true of certain analyses, but it would not afford some of the representations we have seen the state of the art, particularly annotations and visualisations. The same can be said of UTL which allows the calculation of interaction indicators and adding of new user-created events but does not provide for user-created annotations.

It is no doubt possible that some combination of these models or a slight modification of one of them would allow all the forms of analysis found in the state of the art, but, without better knowledge of what analysis in CSCL entails, there is no guarantee that this would be done in

coherent way, or of defining which class of analyses such a model would enable and which class of analyses it would exclude. This is a common theme in computer science models, such as UTL, that do not really adequately describe what analysis problems they can and cannot solve. While the authors of trace-based systems do attempt to define the class of queries and calculations which can be performed, it is very difficult to match this to a class of trace analysis problems because of the link which is missing in understanding how analysis can be supported by such systems.

1.5.3 What can we learn from existing tools

In the state of the art, we have presented a large number of tools and methods, for the analysis of audio and video, for the analysis of digital traces, and for the analysis of a combination of the two. These tools include features such as visualisation, synchronisation, annotation, coding, transformation, abstraction, filtering, calculation of statistics, sequence detection, sequence comparison, sequence characterisation, social network analysis, data mining, etc.

All these features are without doubt of great use for various forms of analysis, but with the notable exceptions of ABSTRACT, MacSHAPA and NXT, none of these tools give us information on the analysis process they are intended to support and, with the further exceptions of DRS and the Session Browser, none provide much information on how such features should be implemented.

1.5.4 Conclusion on the state of the art

This discussion of the state of the art highlights the divide between computer science and the human sciences. Each of these fields is interested in solving its own complex problems (e.g. the theory behind an analysis, the computational ability of a given model), but it seems to be difficult to find goals which are pertinent to both communities when designing analysis software and models. This review illustrates the need for a description of analysis in CSCL upon which a model for representing analyses can be built. In this way, an analysis tool could be created which would not be a list of features, but the implementation of a grounded model of analysis in CSCL.

1.6 Goals of the dissertation

In this section, we are finally able to define the three goals of this dissertation, whose results will be presented in the following chapters. We place these goals within the field of trace engineering and, for each goal we explain the form of the result, how we will evaluate it and its relationship with the results of the other goals.

1.6.1 Trace Engineering

While this dissertation essentially concerns the field of computer science, addressing the issue of capitalising on the analysis of CSCL data from a purely CS point of view would not, as we have shown, fully solve the problem. Indeed, from a purely CS point of view, the problem is not, in itself, definable. We propose to place the work presented within this dissertation in the field of *trace engineering*. This field has not been (to our knowledge) named as such before, although much existing work could be termed to be in this field (e.g. Georgeon, 2008; Settouti *et al.*, 2009). Trace engineering extends the concept of knowledge engineering as explained by Georgeon (2008), and concerns itself with the representation and manipulation of traces and, as such, by the knowledge embodied in those traces. In this case, we consider a trace as being a recorded proxy for what was observable when the trace was recorded. As such, trace engineering as a scientific field aims to build a body of knowledge on concepts, methods, theories, techniques and technologies which can be useful to build tools for the manipulation and representation of traces. Trace engineering must (almost necessarily) be applied to the representation of knowledge from domains outside of computer science (in our case, knowledge particular to the domain of analysis of CSCL situations). Because of this, a traced-based system cannot be considered in isolation and must take into account all the aspects of the various knowledge domains it is applied to.

In the case of CSCL analysis, trace engineering therefore includes developing an understanding of analysis in as much as this understanding can contribute to designing solutions to assist this analysis. It should also embrace the various CS techniques which can help analysts discover and reify knowledge within traces, including but not limited to data mining, document and database engineering, information retrieval, information visualisation, etc.

In this dissertation, however, we will principally concern ourselves with the means by which we can build systems which assist humans in constructing analytic knowledge and representations from traces (as opposed to methods of automatic analysis) and with the ways in which such knowledge can be modelled in order for its reuse to be as easy as possible.

1.6.2 What artefacts are built during analysis?

Our first goal, as suggested in the discussion of the state of the art, is to gain a better understanding of the artefacts which researchers construct, use or manipulate to represent knowledge during the analysis of a corpus. We will do this by examining data analysis from a broader perspective, looking at CSCL data in particular in order to define in what way it is "special" and by considering all the representations which we have discussed in the state of the art as being examples of artefacts which researchers wish to construct.

This will result in the definition of an abstracted artefact which is defined by the operations which can be performed upon it and the types of analysis which can be done with it. This definition will ensure that it does indeed allow CSCL analyses and, following object-oriented design principles, will guide us in later implementation steps.

This object will be evaluated in three ways:

- Can it represent all the analytic artefacts and moves presented in the state of the art ?
- Can the definition of such an artefact through the operations on it allow and inform the design of a model for this artefact?
- Can the analytic challenges presented in our case studies (which will briefly be introduced in §1.6.5 and presented in detail in §5.2) be resolved through the artefact as defined?

1.6.3 How should such artefacts be modelled?

The second goal of this dissertation is to propose a model for this generic artefact which enables the operations defined on it to be applied. We will be able to re-examine existing trace and analysis models in the light of our newly defined artefact in order to examine the modifications needed on existing models. We will present this model as a UML class diagram.

We will also examine more closely the properties of various kinds of artefact and what impact these properties have upon the analysis process. This will allow us to discuss the level of detail at which such artefacts should be modelled in order to provide as much assistance as possible to analysts while requiring as little information from them as possible.

This model will be evaluated according to three criteria:

- Does it afford the operations defined on the artefact defined in the previous section?
- Can it inform the implementation of an environment for analysis through the manipulation of such artefacts?
- Could all the tools presented in the state of the art be implemented using this model?

Note that the first will be true by definition, the second ensures that this model embodies knowledge which can be used for future work in the field of trace engineering and the third extends the validity of the definition of a replayable to ensure that it is indeed not "just a list of features".

1.6.4 Implementing a replayable-based analysis environment

The third goal of this dissertation is to implement an environment for analysis through the manipulation of analytic artefacts. This implementation should confirm that it is possible to implement such artefacts as defined and modelled. It should also inform future work on methods for implementing various features and explore how generic and new features can be integrated with the existing environment.

The implementation will not be evaluated as such, but will serve as a tool with which to conduct case studies for the evaluation of the other results presented in the dissertation. The strengths and limitations of the implementation will however be examined and discussed.

1.6.5 Methodology

Although the goals of this dissertation are presented in a quasi-sequential fashion, it should not be thought that they were achieved in this way. The process of satisfying the three goals was concurrent and iterative, with the definition of the replayable informing the model, the model informing the implementation and the implementation being evaluated through a series of case studies (cf. Figure 1-24). The findings of these case studies in turn served to evaluate

the definition and the implementation itself. As such, the research process continued until we were satisfied with the replayable definition and model with regards to the criteria determined for their evaluation. The development process also integrated aspects of usability and features designed to support the user base constructed during the case studies. In each chapter we will, however, report only the final results, with the results and impact of the case studies being reported in chapter 5.

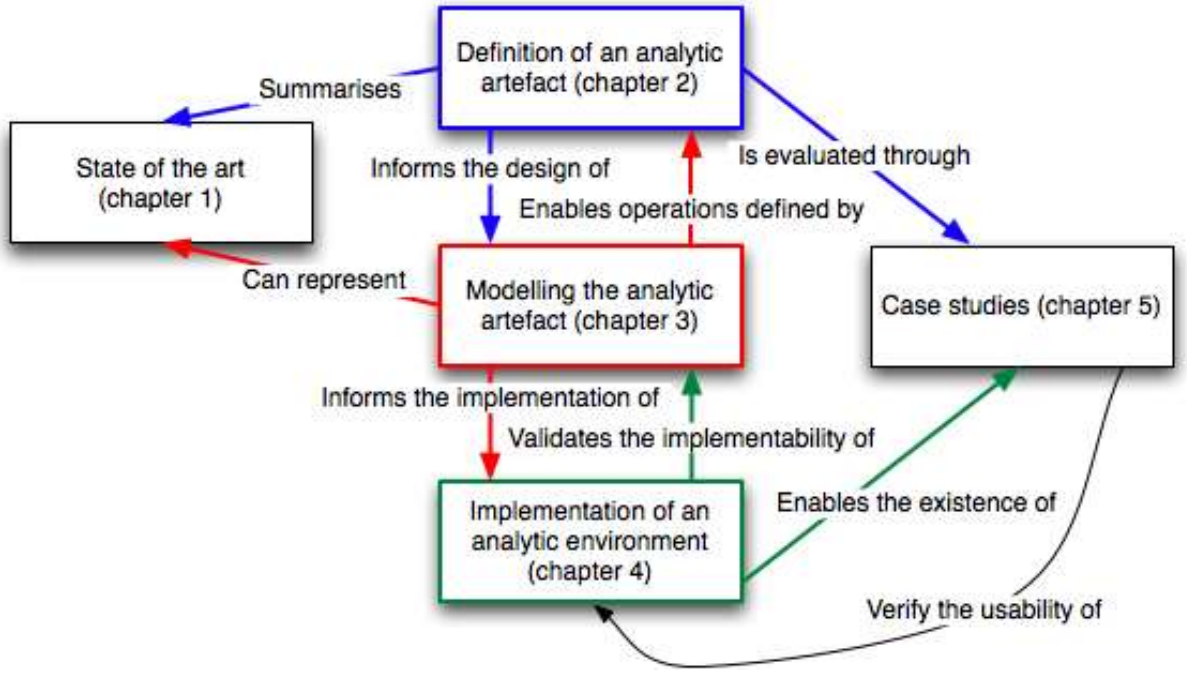


Figure 1-24 Relationship between the three results (center boxes) presented in this dissertation. Each result is evaluated according to the statements in the arrows stemming from that result.

1.7 Conclusion on research issues

In this chapter, we sought to define the research issues to be addressed in this dissertation. In order to define exactly what problems needed solving in order to develop a model for analysis and capitalising of analysis of CSCL data, we first gave an overview of the field of CSCL and of the uses of traces of human interaction. We then examined the state of the art in models, methods and tools for analysing traces of human interaction, with a special focus on the field of CSCL. We discussed the state of the art with regard to three themes: what we know about analysis, models for representing traces and analyses, and the extent to which existing tools can inform us. The major issues raised were that we do not have a clear idea of what analysis

is in CSCL and that therefore, it is difficult to evaluate whether existing models are appropriate or not. The lead us to define three goals for this dissertation:

- Understand the kinds analytic artefacts created during analysis, define a generic artefact created for representing analytic knowledge and render explicit the kinds of analyses it can assist.
- Model this artefact and highlight its properties which are relevant for analysis.
- Implement an environment for manipulating the artefact defined by the results of the first two goals and which serves as a tool with which to evaluate these results through case studies.

Finally, we briefly addressed the methodological issues, showing how the three goals are interrelated.

2 Defining an analytic artefact

2.1 Introduction

In this chapter, we present the first result of this dissertation, showing how we construct an artefact which covers a wide range of the artefacts already constructed by researchers during analysis of CSCL data. With regard to these artefacts, three questions can be asked: why did researchers choose to construct such artefacts? What artefacts should the researchers construct? And how can researchers construct these artefacts? The first question is epistemological, the second methodological and the third is pragmatic. We will attempt to focus on the first and third questions, rather than address the methodologies which exist and are so numerous (potentially infinite) as to be beyond the scope of this dissertation.

We first examine analysis from a broad perspective in order to establish a general analysis process. We discuss various epistemological traditions and take a closer look at the specificities of CSCL data in order to define the kinds of analyses which this process can and cannot address. We then draw upon a wide range of examples to examine the kinds of artefacts which are created, manipulated and transformed during analysis, This enables us to define a generic analytic artefact we call a *replayable* and describe the operations which can be performed upon it, transforming it into a new artefact of the same kind.

2.2 What is analysis?

According to Thomas & Cook (2005), analysis is an iterative process which goes through a knowledge crystallisation (or sense-making) loop, whose steps are the gathering of information, the representation of this information in a way which assists interpretation, the development of new insight through the manipulation of this representation and the creation of a new knowledge object which *crystallises* this new insight. They claim that analysis should be supported by *analytic discourse*, defined as the "technology-mediated dialogue between an analyst and his or her information to produce a judgment about an issue" (p. 38). This discourse creates a series of *reasoning artefacts*, which can range from simple pieces of raw data to complex models abstracted up from the raw data and which can contribute towards making defensible judgments about an issue. Some of these *reasoning artefacts,*

termed *products*, can be thought of as artefacts which are specially designed to be shared with others.

Russel, Stefik, Pirolli, & Card (1993) define sense-making as the "process of finding a representation and encoding data in that representation to answer task-specific questions" (p. 1). They describe sense-making as a *learning loop complex* (cf. Figure 2-1), here destined to make sense of a task structure, but which can be adapted to making sense of other forms of data. The learning loop complex starts by the search for a good representation. Data is then encoded in that representation and residue (items which are poorly represented by a given representation) are identified, giving rise to a representational shift (the search for a better representation which also includes the residue). The result of this loop is a representation of the essence of the data encoded therein.

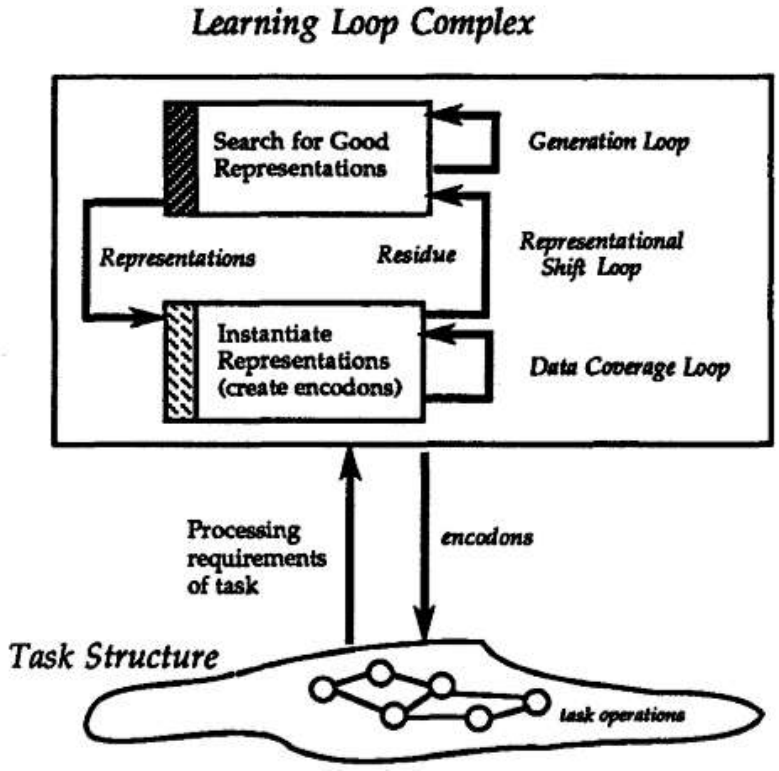


Figure 2-1 Sense-making is finding a representation that organises information in an optimal way (with regard to various forms of cost). The result of the learning loop complex is a representation and a set of encodons (data encoded in that representation). (Russel *et al.*, 1993, p. 2)

The notion of sense-making can also be found in the work of Tesch (1990) on qualitative analysis. Researchers analyse their corpus by removing extracts from their context which they then compare and re-associate. The extracts are then given a new structure which encapsulates the meaning the researcher wants to give to them. For example, in a series of interviews, all

answers which describe the benefits of a certain product might be isolated, compared with each other to abstract a complete list of benefits. Each benefit from the final list might be illustrated by a quote or set of quotes pertaining to that benefit.

We have already given an overview of Exploratory Sequential Data Analysis (ESDA, cf. §1.4.5.1), which describes analysis as a double loop. The outer loop consists of asking questions and answering them within an epistemological framework which defines questions which can be asked, the data which can be collected and the kinds of answers which are appropriate. The inner analytic loop is the iterative process of creating products from the collected data. Sanderson & Fischer (1994) define the operations as the *eight Cs*: chunks, comments, codes, connections, comparisons, conversions and computations. Sanderson & Fischer (1994) further explain various epistemologies (the "formal concepts" which guide the analysis) which use ESDA. They identify three "intellectual traditions" (or collection of research practices which inherit some common fundamental assumptions), each of which provides different answers to the questions "what is the issue at hand?", "what should be observed?", "what operations should be done?" and "what is an acceptable type of answer". The following paragraphs are paraphrased from Sanderson & Fischer (1994), p. 270.

The behavioural tradition (not necessarily *behaviourist*) emphasizes objectivity and quantitative analysis and often uses hypothetico-deductive tests in order to achieve objective results. It observes data in the field to ensure ecological validity or in a laboratory to control variables and attempts to observe subjects belonging to a broad sample to ensure statistical validity. The focus of analysis is on objectively defined behavioural observations, which are analysed using formal coding schemes and statistical analysis. The results are of a quantitative nature, and should ideally be replicable generalisations.

The cognitive tradition seeks to model cognitive processes. It tends to focus as much on laboratory experimentations as field studies and considers individual verbalisations as evidence for cognitive processes and structures. Because of the study of verbalisations, it is often less easy to conduct extended studies, which can constrain sampling adequacy. For the same reason, while coding is a method of choice for analysis, the objective attribution of categories is more difficult and is highly dependent on the context in which utterances are produced. Models of cognitive processes are then constructed or evaluated with the collected data. The results typically report the extent to which the data conforms to existent or newly developed models of cognition.

The social tradition attempts to provide accounts of social phenomena using empiric and ethnographic methods. It uses field observations and, mindful that the presence of the observer will influence the situation in any case, emphasizes the value of the observer's participation. All forms of social interactions are considered valid objects of study, including utterances, gestures and actions. In this tradition, the coding and annotation of the data "is" the interpretation, appearing more like final research statements than data points. Analysis is more frequently qualitative than quantitative, emphasising a plausible and robust account among the many possible accounts of the data.

From the perspective of analysis in CSCL, the recent work of Suthers (Suthers, 2006; Suthers, Dwyer, Medina, & Vatrapu, 2007; Suthers & Medina, 2008) proposes the notion of a *contingency graph*: a generic representation of data which can be adapted to various CSCL epistemologies and methodologies (according to their specifics) and can serve as a *boundary object* between these epistemologies. This contingency graph (cf. Figure 2-2) describes vertices which represent *media coordinations* (actions taken by participants in a technological environment; the means through which participants coordinate between personal and public realms (Hutchins, 1995; cited by Suthers & Medina, 2008)), and the arcs between these vertices are the *contingencies* which describe the objective relationships between the various media coordinations (note that this structure is reminiscent of that used by ABSTRACT described in §1.4.1.5).

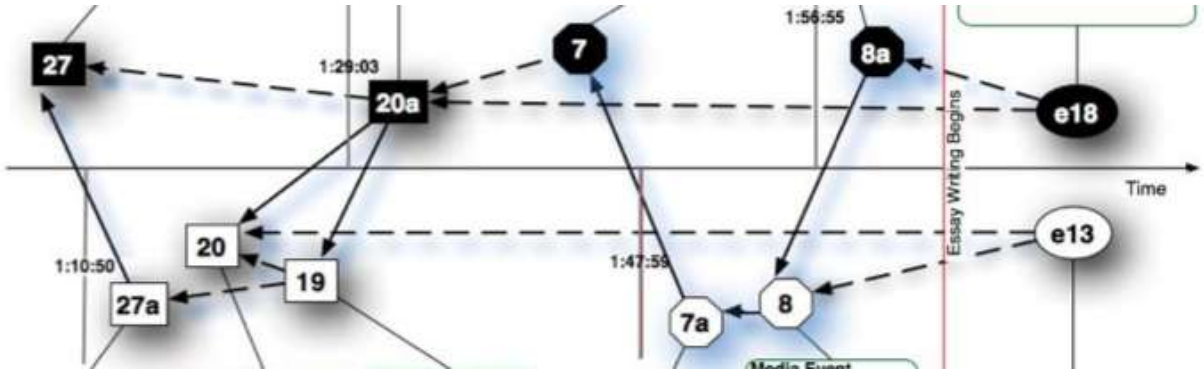


Figure 2-2 A contingency graph illustrating the dependencies between the contributions of two students (respectively top and bottom) in order to trace the transfer of knowledge between the two. (Suthers *et al.*, 2007)

From these contingencies, which can be temporal dependencies, inherent media dependencies (e.g. a forum's reply structure) or dependencies of a semantic nature (re-use of a phrase, word or meaning), the analyst is able to construct interpretations of the intersubjectivity present in the data. With this representation, Suthers and Medina explain that it becomes possible to

follow the trajectories of the actors, the ideas they express and the artefacts which they construct. In turn, these trajectories can be used to trace the movement of one type of entity (actor, idea, artefact) with respect to the other two. Confluences occur when trajectories intersect (e.g. an representational notation is taken up by a different actor, or an idea is expressed through a different artefact). These confluences, Suthers and Medina argue, are the places to look for transformations which may be indicative of certain types of learning or meaning-making.

Suthers and Medina (2008) explain that the contingency graph is designed to answer a number of analytic challenges within the field of CSCL.

- The first challenge is that data is distributed across media (multiple audio, video, and digital traces), space and time (in distant asynchronous communication, different actors may be aware of different media coordinations at different times). The contingency graph answers this problem by gathering all the data into one analytic artefact. However, Suthers and Medina highlight that more work needs to be done on visualising and querying this graph in useful ways.
- The second challenge stems from the contingent nature of human behaviour. In other words, human action depends on the context in which it occurs (the physical environment and recent history of interaction). Contingency graphs represent these contingencies explicitly, affording both automated and human judgment of the pertinence of these contingencies during analysis. Compared to methods of analysis in which these contingencies are not explicit, the contingency graph almost makes the situation worse, forcing analysts to deal with their complexity. Furthermore, Suthers and Medina explain that some aspects of the context are not included in the media coordinations (e.g. because they occurred before the recording) or are non interactional (such as prior knowledge).
- The third challenge lies in interpreting non-verbal behaviour such as moving artefacts in a co-constructed representation. While these actions can at least be included in the contingency graph, interpreting their significance (and not letting them drown out other, potentially more significant, media coordinations) remains a challenge.
- The fourth challenge is related to the problem of scaling up interaction analysis to larger and more numerous data sets in order to increase the generalisability of findings. How can methods be found which construct a meaningful analysis of such large data sets and allow analysts to focus their attention on selected phenomena

without omitting important data? This problem is addressed in contingency graphs by affording goal directed tracing of the objects under study (e.g. tracing back from a particular learning outcome). Suthers & Medina note that automatic tools for filtering and finding relevant pathways in the contingency graphs will make this process easier and that new tools need to be developed to enable querying, filtering and visualisation.

- The fifth and final challenge is that of addressing multi-scale phenomena (at different levels of granularity: individual, group, etc.) and to describe the relationships between phenomena across scales. Suthers & Medina express the hope that contingency graphs will facilitate use of computational methods to find patterns at larger scales.

Strijbos & Fischer (2007) survey methodological challenges in CSCL and also list multi-level analysis and the difficulty of integrating non-interactional contextual elements to the analysis. They add three further challenges. The first is the fusion of hybrid methodologies: this requires a deep understanding of all methodologies being used in order that their fusion be coherent. They explain that this "is not a task for the 'lone researcher', but rather for a research group or even several research groups in collaborations" (p. 392). The second is that of using segmentation and coding in a reliable and valid way. They highlight the problems of defining a coding scheme and the rules for applying the scheme, and using coding by two researchers to measure reliability. Finally, they highlight the difficulty in assessing concepts such as divergence, convergence and the sharing of knowledge through interpretation of online discussion.

2.3 A generalised analysis process

Based on the above accounts of analysis, we propose that analysis can be described as a loop wherein analysts gradually augment their initial data pool with artefacts which give an increasingly detailed, rich, abstracted or appropriate (to further understanding) account of the situation under analysis. At the end of this process, these artefacts not only *reify* or *crystallise* the analytic knowledge generated during analysis, but *embody* the interpretation or meaning which the analyst chooses to give to this data – or finds within it (cf. Figure 2-3). There are two reasons for which we do not think it pertinent to describe this process in greater detail. First, as described by several authors (cf. 1.5.1), more flexibility implies greater analytical expressivity whereas a more constrained process would limit the possible analyses. Second, a more detailed process would require assessing how the artefacts created at each stage of the

process are different from each other and possibly creating different models for each. In other words, in basing our support for analysis on such a simple process, we hope a) to save time and effort by minimising the number different kinds of artefacts we need to model and b) to provide more power and expressivity to the analyst.

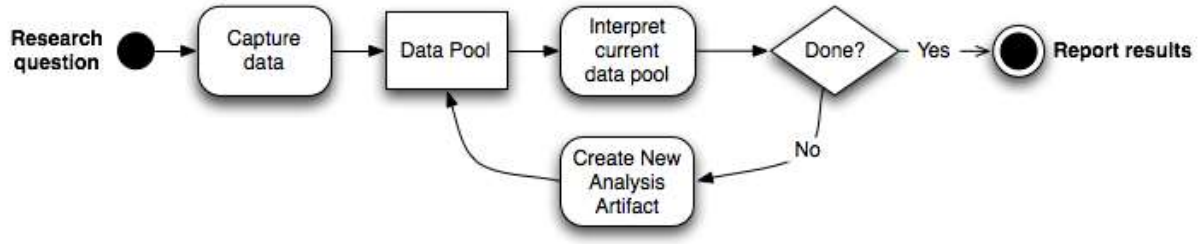


Figure 2-3 The proposed model of the analysis process

Our challenge now lies in examining the artefacts which are created during the analytic process and determining whether they can be abstracted into a generic artefact which describes them all. If so, this artefact should be able to represent the original data and the operations defined on this artefact should enable the creation of all subsequent analytic artefacts.

2.4 The specificity of CSCL analysis

Before attempting to define such an artefact, we must first determine its field of applicability – in other words, to what kinds of analysis it is suited, and to what extent it needs to be specifically designed for the analysis of CSCL data. As we have seen, the field of CSCL can be decomposed into the words that constitute its name: computer support, collaboration and learning. Let us examine each of these components in order to see what specificity they bring to analysis and whether they differentiate analysis in CSCL from analysis of collaborative learning, technology enhanced learning or computer-supported collaborative work.

Computer support entails that the resulting traces are digital and therefore produced with a tracing intention (by the designer of the tracing mechanism) and are given meaning by the environment in which they are traced. For such a trace to be useful to an analyst, the tracing intention must be relevant to analysis – or better still specifically designed for it – and the implicit knowledge about the environment which is embodied in the trace must be known to

the analyst. Such traces are particularly rich, since computer mediation "transforms communication into substance" (Dillenbourg, 2005), rendering linguistic interactions explicit and directly treatable by automatic or manual means. One might also say that computer-mediated interactions are "already transcribed". In the case of other forms of computer mediation (graphical workspaces, etc.) this richness can also translate into an increase in difficulty for actions to be understood by the researcher. Once tools are designed to produce traces, very little extra effort is required for recording, affording the ability to produce recordings of interactions over a long time span incorporating both synchronous and asynchronous interactions in multiple media. Many situations combine face-to-face and computer-mediated interactions (LEAD, 2006) and require video analysis to understand them fully. In spite of the richness of digital traces, we cannot count on CSCL data consisting exclusively of digital traces. They do, however, add large quantities of data, which are not necessarily easy to understand.

Collaboration is interesting because of the human interactions which it entails. These interactions, as Suthers & Medina (2008) explain, are particularly contingent on the context in which they are produced and must therefore be reproduced in context to be understood. Furthermore, as computer mediation creates persistent inscriptions of communication, the context is not uniquely temporal (i.e. the nearest context is not always the events which happened immediately before, as in the case of non computer-mediated communication or even non collaborative human-computer interaction, but can span over days, months and years) and thus requires the adaptation of various forms of discourse and content analysis to extend their context further than single utterances and adjacency pairs.

Finally, we address the aspect of learning. As shown by the MULCE project (cf. 1.4.1.3), a learning corpus contains elements to explain the pedagogical scenario and the prior knowledge and abilities of the students, making the context even more important. In particular, the pedagogical scenario can serve as a yardstick against which to "measure" the process which actually unfolded. Otherwise, aside from situations which include assessment (e.g. educational data mining), the definition of what constitutes learning and how it interacts with computer-mediation and collaboration will depend on the epistemology in which the analysis is performed. Reimann (2009) and Koschmann (2002) advise that a prime object of study for CSCL should be the interactional process. As explained by Suthers (2006): "The focus [of CSCL] is defined by what aspect of human collaborative activity we examine and try to make sense of […]." In other words, it is up to each researcher to define which aspects

of activity constitute learning according to their epistemological beliefs and thus to define the focus of their study.

This point must be taken into account. Indeed, analysis tools "influence the researcher's logic" (Savoie-Zajc, 2000; cited by Teutsch *et al.*, 2008). For an object to be generic, we must minimise the decisions we make on behalf of the analyst. For this reason, we do not make any epistemological or methodological commitments beyond the following:

- The researchers' epistemology leads them to collect traces recording the *activity* of the subjects under study;
- These traces may be digital but may also incorporate audio, video or any other record of the activity;
- The recorded activity is some form of *situated human interaction* (an interaction between a human and the environment in which he finds himself, which may include other humans).

In other words, the range of epistemologies we intend to address are those which lead researchers to collect traces of situated human interaction. These epistemologies will (at least partially) define methodologies for the analysis of that data. Assuming our state of the art covers the variety of operations performed when executing these methodologies, the object we define should allow these methodologies to be carried out.

Exactly what researchers wish to study will define the aspects of activity which are of interest to them, whether it be learning, collaborative learning, collaborative work or any other kind of activity. We make the claim that, for the purpose of computer support for analysis, CSCL analysis is not of a different kind than other analyses of situated human interaction. However, the difficulty of showing how learning unfolded, combined with the complexity of collaborative interactions and the richness of digital traces, increase the challenges faced in other kinds of analyses and focus the analysis all the more on the activity itself. For this reason, a tool able to support the analysis of complex CSCL situations should also be suited for the "simpler" analysis of other situations. It may also be because of this complexity that CSCL has demanded such rigour in defining computer support for its analysis (although, as the tools in the state of the art have shown, CSCW, HCI and other fields involving the analysis of activities where order and chronology are important have also required and motivated the creation of powerful computer-support for analysis).

2.5 Overview of operations on analytic artefacts

In this section, we synthesize the features of the various tools presented in the state of the art (to which we add spreadsheets – which are commonly used for coding – Suthers' contingency graphs and common audio and video media players). We organise these features by themes (slightly adapted from Hilbert & Redmiles (2000) cf. 1.4.5.2) whose purpose for analysis we will explain and from which we will later extract the operations which can define a common artefact for analysis: these themes are *contextualisation*, *visualisation*, *transformation*, *enrichment*, *comparison* and *aggregation*. In order to compare tools, their features with regard to the first four of these themes are listed in Table 1.

In this table, we can note that none of these tools present all the kinds of visualisation, transformation and enrichment which we have reported. Furthermore only a limited subset of the tools (NXT, MacSHAPA, ActivityLens, DRS and ABSTRACT) present non-static forms of contextualisation (replay and synchronisation) and features in each of the other columns. Even in these tools, the presence of both automatic and manual features for transformation and enrichment is lacking.

2.5.1 Contextualisation

One of the important necessities we have mentioned, linked to the contingent nature of the data, is the need for *contextualisation* or re-contextualisation in order to present the data to the researcher under a form which both renders the events themselves understandable (i.e. show what a trace event means) and affords the understanding of the pragmatics of the situation (i.e. allows the researcher to interpret this event in context). Contextualisation is part of the connections aspect of the eight Cs and mostly matches the synch and search theme presented by Hilbert and Redmiles.

The first aspect of contextualisation relates to the fact that traces record data which is positioned in time. Almost all the tools in the state of the art attempt to restitute the temporal dimension of the data, whether it be through visualisations which present time on a vertical or horizontal axis or by providing the means to "replay" the data via a control with "play" and

"pause" buttons in the case of tools which allow video to be replayed (or in the case of tool replayers),.

	Contextualisation	**Visualisation**	**Transformation**	**Enrichment**
NXT	Replay, Synchronisation, Visualisation proximity	Graphical timeline, Text	Segmentation, Filters (query language), Sequence search	Annotations, Categorisation, Collections, Structures
SAW	Synchronisation, Visualisation proximity	Graphical timeline, Table	Filters, Event Insertion	-
CORDTRA	Visualisation proximity	Graphical timeline	-	-
Travis	Visualisation proximity	Graphical timeline	Filters	-
ViCoDiLi	Visualisation structure	Tree	Filters	Collections
LISTEN session browser	Visualisation structure	Tree	Filters	-
MacSHAPA	Replay, Synchronisation, Visualisation proximity	Graphical timeline	Filters (query language), Event grouping, Sequence search	Categorisation, Annotation
ActivityLens	Replay, Synchronisation, Visualisation proximity	Table	Filters, Event grouping	Annotation
Replayer	Replay, Synchronisation, Visualisation proximity	Graphical timeline, Time plot, Map	-	-
DRS	Replay, Synchronisation, Visualisation proximity	Graphical timeline, Table, Time plot	Segmentation	Annotations, Categorisation
ABSTRACT	Replay, Synchronisation, Visualisation structure	Graphical timeline, Graph	Filters (query language)	Connections
Tool Replayer	Replay	2D image	-	-
KSV	Visualisation structure	Graph	Filters	-
Social Network Analysis	-	Graph	-	-
State graph	Ordered Vertices	Graph	-	-
TagHelper	Visualisation proximity	Table	-	Automated Categorisation
Spreadsheet	Visualisation proximity	Table	Row insertion	Adding columns
Video player	Replay	2D image	-	-
Audio player	Replay, Synchronisation	Time plot	-	-
Contingency Graph	Visualisation structure	Graphical timeline	-	Structures

Table 1 Summary of the kinds of contextualisation, visualisation, transformation and enrichment which can be used in various tools to create and exploit analytic artefacts.

When a tool allows the simultaneous visualisation of several artefacts which represent, for example, the same data in different ways or co-occurent data in different media or modes, it often allows the visualisations to be synchronised in time (e.g. Replayer, cf. Figure 1-22). This synchronisation affords the combination of multiple views in order to better understand a situation. Furthermore, some artefacts can serve as an index for others. For example a transcription can assist the navigation through a video. Finally, the iterative nature of analysis leads to increasingly abstract artefacts (which describe phenomena at different levels, such as the individual utterances, their coding into interaction categories and a separation of the whole activity into phases). Synchronisation is one way of maintaining a link between the different levels of abstraction and with the primary data in order to confirm the validity of the knowledge crystallised from secondary artefacts[7].

Another form of contextualisation is associated with the inherent structure of the media that were used. Whereas an oral conversation mainly follows a chronological order where temporal proximity is the strongest contextual element, this is not necessarily the case for communication which is mediated by tools such as forums or argumentation graphs. This structure can be reconstructed by tools such as ViCoDiLi (cf. §1.4.4.2), which is designed to re-establish different kinds of forum structures. In other cases, events can exist within a hierarchy of events, as in the case of the data for which the Session Browser (cf. §1.4.2.1) is designed.

Last, as the MULCE project (cf. §1.4.1.3) shows, the context in which data was collected (pedagogical scenario, interviews, data about the participants, etc.) must not be forgotten and is often indispensable for analysis.

2.5.2 Visualisation

We now describe the variety of forms of visualisation which can be observed:

- Two dimensional images (which tend to change over the course of time), used by video players, the DREW replayer and Replayer's map view.
- Spreadsheet objects with rows and columns (MacSHAPA, ActivityLens, DRS).
- Graphical timelines which present each event in a symbolic form (SAW, CORDTRA, MacSHAPA, ABSTRACT, TrAVis).

[7] The terms *primary data* and *secondary artefact* are used here in a very pragmatic sense: primary data is the trace that was recorded and all artefacts which are created from this data are "secondary". For a discussion of this kind of distinction, see for example Mondada (2006).

- Timelines which allow users to select and save time intervals (NXT, DRS).
- Curves which plot a metric value (a value that has been measured as a certain quantity) as it varies over time (DRS, Replayer and audio players which offer a view of the intensity or pitch of an audio signal over time).
- Text on which it is possible to add anchors (NXT).
- Trees, most often presented as tree tables : tables in which each line is also the node of a tree and whose children are shown, indented, in the rows below (ViCoDiLi, Session Browser).
- Graph structures showing the links between certain objects such as events, documents or actors (transition graphs, social network analysis, KSV, ABSTRACT, contingency graphs).

The majority of these visualisations contribute to contextualisation, whether it be by representing structure or time. The representation of time can take several forms. Some artefacts only show a "freeze frame" at a given time which does not provide any indication of temporal context unless replay is involved – in other words, they must be animated. Audio players are an extreme case of this – when pause is pressed, the listener knows nothing of what is happening and the audio "representation" cannot exist outside of replay. Often, sequential events are visualised in a sequential way, as consecutive lines in a spreadsheet or through adjacent symbolic elements in a graphical timeline. Finally, when a visualisation shows events, but does not indicate temporal proximity through spatial proximity, the numbering of these events or their ordering in a graph structure can lead to a notion of sequentiality (e.g. Blake & Rapanotti, 2001). In all the cases where events are represented in a visualisation, they can serve as navigational items for the synchronisation between artefacts through brush and link (cf. §1.4.5.5).

The purpose of these different forms of visualisation is also to present data to analysts in such a way that the aspect of activity of interest to them is easy to see: "the simple act of placing information on a timeline or a map can generate clarity and profound insight" (Thomas and Cook, 2005, p. 37). Appropriate visualisations present many advantages (Card, Mackinlay, & Shneiderman, 1999):

- Increased cognitive resources: human gaze is able to take in large amounts of information at a time; visualisations can serve to expand the working memory and can store large amounts of information in a readily accessible way.

- Reduced search: information which needs to be accessed together can be grouped together; large amounts of data can be represented in a small space.
- Enhanced pattern recognition: using recognition rather than recall[8]; benefiting from abstraction and aggregation of data; using various visual schemata to show relationships (proximity, lines, etc.); showing values, relationships and trends.
- Perceptual inference: using appropriate visualisations can enable easy perceptual inferences
- Perceptual monitoring: large numbers of graphical objects can be monitored if "obvious" features (appearance, motion) allow them to stand out.
- Manipulable medium: computer-assisted visualisations can be manipulated, allowing users to explore the visualisation parameter-space, as opposed to static visualisations.

Visualisation can be viewed as a special form of conversion (with regard to the eight Cs).

2.5.3 Transformation

The act of transformation takes an artefact (or several artefacts) made up of a certain number of elements (for example events) and produces a new artefact composed of additional events, a restricted number of events or events at a different granularity. This covers several aspects of the eight Cs: chunks, constraints, connections, and conversions. Transformations can be manual, assisted by a tool or automated, according to the availability of a tool to assist the transformation and to the capacity of the state of the art to automate the transformation without human intervention.

Tools which allow the creation of chunks include all the tools which allow the selection of an interval on a graphical timeline (NXT, DRS) or to select and subsume a set of events (ActivityLens, MacSHAPA). Conversion capacities are similar and are often automated as in the case of sequence search – and the creation of new events to represent the identified sequences – found in tools which allow complex queries such as ABSTRACT, MacSHAPA and NXT. Hilbert and Redmiles (*op. cit.*) call both of the above (chunks and conversions which are not visualisations) *abstractions*. More simple queries impose constraints on data such as filtering and feature removal (*slicing* according to Mostow and Beck (*op. cit.*), cf. §1.4.2.1) or the selection of a limited interval (*segmentation* according to Mostow and Beck)

[8] Recognition is well known for being easier than recall, consider the relative difficulty of answering a recall question ("what is the name of that man?") and answering a recognition question ("is that man called Frank or John?")

both of which are called *selection* by Hilbert and Redmiles. Finally, *connections* are created in fusion transformations which combine data from different sources in a single artefact (e.g. in SAW or in contingency graphs), be it by sorting by time or by exhibiting other connections (threaded structure of a discussion tree for example).

In this description, conversions which only consist in visualisation of the *same* artefact in a different way, fall under visualisation, not transformation (as they don't modify the elements which make up the artefact). Furthermore, as transformations take an artefact and produce a new one of the same kind (the one we are defining in this chapter), the *recoding* aspect described by Hilbert and Redmiles (i.e. the fact that the result of an abstraction is then "recoded" to create a new data stream which can be further manipulated) is implicitly understood to be present.

2.5.4 Enrichment

In the knowledge crystallisation loop, the generation of new knowledge is followed by its inscription in new artefacts. Apart from creating new elements, this inscription can also be done directly "on" already existing elements. These enrichments can take the form of *comments*, *codes* and *connections*. In video annotation tools in particular, these enrichments are directly added onto time intervals. In other tools, they are added to events (or to rows in the case of a spreadsheet).

Comments (or annotations) allow researchers to reify some pieces of knowledge, in order to store them as a future reminder, to share them with other researchers or to construct their interpretation of the data. Annotations can also be used in an exploratory way as a preliminary for constructing a set of categories or a controlled vocabulary, for example in the case where the same annotations are often used.

Categories or codes serve the purpose of "abstracting" the data to a limited vocabulary upon which statistics are easier to calculate and patterns easier to find. Going further than controlling the vocabulary, tools such as NXT allow the creation of categorisation ontologies. This serves three purposes: ontologies serve to describe and document the categorisation scheme, can be used to impose rules on how multi-level categories can be applied and allow the construction of queries on data with complex annotations.

In the state of the art on audio-video analysis, the structures which are imposed by tools on annotations, codes and categories (cf. §1.4.3) are often discussed with much criticism. Indeed,

on one hand, they impose on the user of the tool a complex mental model which they will have to understand and on the other hand they can impose a certain way of doing things (e.g. building up the complete coding scheme before applying it). It should further be noted that sometimes these constraints are unavoidable. Their absence is not necessarily an implicit liberty for analysts to create the structures appropriate to their needs but can be an explicit simple structure (e.g. an event only being able to have a single category) – in other words, an apparent lack of constraints may actually be a simple constraint which is as much of a constraint as explicit complex structures. Coding is such a ubiquitous activity in digital trace analysis (particularly in the case of online discussion) that several tools attempt to automate this coding with machine learning techniques (e.g. TagHelper, cf. §1.4.2.1).

A similar function to coding is the creation of collections of elements of like nature. In ViCoDiLi, this is done by adding elements to a "basket". In video annotation tools, particularly in the case of Conversational Analysis (Sacks, Schegloff, & Jefferson, *op. cit.*), this is done by adding a keyword to an element, which acts as a *tag*, a form of keyword that is used for manually indexing a corpus. A collection can be seen as the result of a query for all elements with a given tag.

One particular type of enrichment is the manual or automated expression of structure between elements (e.g. NXT). This can serve to crystallise knowledge about relationships and to create powerful visualisations and queries based on these relationships. As Suthers & Medina (2008) explain, these structures can then be followed to trace actors, ideas and artefacts (cf. §2.3). This notion of structure can also be found in KSV and transition graphs, and is exploited in social network analysis. Structures of this sort can also be found in work which seeks to explain the contingencies between different events (e.g. Lund *et al.*, 2008; Prudhomme, Pourroy, & Lund, 2007).

Enrichment is conspicuously lacking in the themes presented by Hilbert and Redmiles. It could conceivably be addressed by transformation in the form of abstraction, but we believe that doing so is a way of bypassing the lack of enrichment possibilities and a misappropriation of transformation (we will explore the advantages of this distinction further in the following chapter).

2.5.5 Comparison

Of the eight Cs defined by Sanderson and Fischer, two have not yet been addressed and are absent from the comparison between various tools presented in Table 1. The first is comparison. These comparisons can take several forms. If we decompose analysis into the constituent components of the methodology for analysis, the researcher applying this methodology and a corpus involving (in the case of CSCL) a set of learners, computer artefacts and pedagogical scenarios under study, the variation of just one of these parameters can illustrate its role or be used to confirm results. Applying the same coding to the same corpus by different researchers is used to validate a coding through inter-coder reliability measures (De Wever *et al.*, 2006). The parallel or sequential application of different methodologies can lead to new results or to confirmation of results by triangulation (Suthers, 2006). Last, the comparison of groups, individuals, tools and experimental protocols can be used in contrasting case studies or to obtain statistical information, yielding qualitative and quantitative results, respectively.

In spite of the obvious necessity of comparing various analytic artefacts, few tools explicitly support this operation. Georgeon (2008) notes that comparison is the only one of the eight Cs he hasn't felt the need to implement. The tools Hilbert and Redmiles report as supporting sequence comparison, such as MacSHAPA do this in part but this only allows certain kinds of comparisons (those which are both quantitative and attempting to compare the sequential process). The designers of SAW mention the need for researchers to collaborate, allowing them to share analytic artefacts and partial results (any software which lets users save a file enables the sharing of this file – but we haven't seen further mention that this is intentional). At best, to compare two artefacts, they could be opened side by side (a form of visualisation) or evaluated statistically (a form of transformation). This highlights one of the reasons spreadsheets are so popular: user-defined statistics are inbuilt and comparing two researchers' codings is as easy as copy-pasting one next to the other (a form of enrichment).

2.5.6 Aggregation

The last of the eight Cs to be examined is *computation*, which Hilbert and Redmiles call *counts and summary statistics* but which probably also applies to *sequence characterisation*. The result of these computations we will call *aggregations*, to highlight the different nature of the artefact they produce from all the other artefacts covered in this chapter. A typical

computation takes an artefact composed of sequential "timestamped" data and *summarises* it, producing a set of numerical values (or in the case of sequence characterisation a transition graph) which applies to the whole duration over which the artefact is defined (e.g. number of turns, number of turns per actor, collaboration ratio, etc.). For example, most kinds of interaction indicators (cf. §1.3.2) are aggregations. Because of aggregations' fundamentally different nature, we will consider them mostly out of the scope of the work presented in this dissertation. They are not temporally situated, do not need contextualising and cannot be visualised in the same ways as other artefacts we have discussed. Aside from the computation itself which leads to the construction of aggregations, the subsequent use of most aggregations (e.g. comparison between aggregations produced from two different groups) can be done in spreadsheets or dedicated statistical analysis software.

We can, however, examine the relationship between aggregations and the data we have termed *metric* (one or more measures which are regularly sampled over time). Discrete time signal processing often uses the notion of a sliding window: a window of a certain breadth (e.g. one second) and shape (e.g. triangular) slides along the signal by successive increments (e.g. a tenth of a second); for each "step" of the window, a computation is performed; the result is a metric data set with one value for each step. Each data point aggregates information in the neighbourhood of the centre of the window. In our example, this would produce one data point every tenth of a second, aggregating data half a second before and half a second after, weighted by the triangular shape of the window (nearby data is counted as more important than more distant data).

This sliding window system could be adapted to consider how an indicator varies over time. Such computations would then produce an artefact similar in nature to the others and would extend the notion of transformation. Such operations are similar to derivation or integration, showing respectively rates of change and the cumulative effects of these changes.

2.6 Definition of a replayable

After defining the types of analyses we wish to support and having examined the themes of contextualisation, visualisation, transformation, enrichment, comparison and aggregation, and related them to the operations and artefacts found in the state of the art regarding analysis (most notably Sanderson and Fischer's eight Cs and the overview given by Hilbert and

Redmiles) we are now in a position to define a generic artefact to support the analysis process presented in §2.3.

We call this artefact a *replayable*, as it can be *replayed* and *synchronised* with other like artefacts. It can also be *visualised* in a variety of ways. These first three operations are ways of presenting a replayable or set of replayables in an understandable way to a human. Some of the artefacts we examined in this chapter cannot be replayed and synchronised in their current incarnation (e.g. contingency graphs). This is rather a technical than conceptual limitation and illustrates all the more the necessity of defining a framework within which the full potential of these artefacts can be exploited. Replayables can also be *transformed* into new replayables or *enriched*, leading to an improved replayable. Finally, computations on replayables can also produce *aggregations*, which are no longer replayables (and can thus no longer be subject to the operations which apply to replayables).

2.7 Conclusion on defining an analytic artefact

In this chapter, we examined analysis from a broad perspective, defining it as cyclic process of knowledge crystallisation wherein analysts iteratively create artefacts which are increasingly rich, abstract or appropriate to further use. By examining the variety of artefacts described in the state of the art and their means of creation through the themes of contextualisation, visualisation, transformation, enrichment, comparison and aggregations, we were able to define the concept of a *replayable*, a generic artefact which can be replayed, synchronised with other artefacts, visualised, transformed and enriched.

In order to define the kinds of analyses such an artefact can cover, we addressed in detail the issue of CSCL analysis and concluded that, in spite of increased difficulty due to the complex multimodal nature of CSCL interactions, it was no different in nature from analysis of other kinds of situated human interaction. We isolated the key epistemological constraint imposed by the use of replayables for analysis as being that researchers have chosen to record and analyse the process of interaction through traces of activity. Exactly what aspect of this activity is of interest and how that aspect can be identified, we leave up to the researcher.

3 Construction of a model for replayables

3.1 Introduction

Having defined the notion of a replayable through the operations which can be performed on it, we now present a way of modelling replayables in order to allow these operations to be carried out. We also explore the relationship of this model with existing trace models and examine in greater detail the different properties that replayables can have and the consequences for their analysis, modelling and implementation.

In this chapter, we first briefly explore what can be meant by the word "model". We then establish the difference between a replayable and its visualisation, and explain the model for replayables we have constructed. This will lead us to discuss the various properties of replayables and their visualisation as well as the consequences for transformation and enrichment.

3.2 What is a model?

In this section, we draw extensively on Kühne (2006). While his work on models is not necessarily the first or the most complete, it serves as a basis to describe the essence of what needs to be said about models in this dissertation. Kühne defines the following:

> *A model is an abstraction of a (real or language-based) system allowing predictions or inferences to be made. (p. 2)*

He cites Webster's new encyclopaedic dictionary (1994) to establish that while a model can be the abstraction of an existing system (descriptive model), it can serve as a prescriptive model for an as yet non-existent system. Stachowiack (1973) defines three features a model must have: a mapping feature, a reduction feature and a pragmatic feature. The first establishes that a model must exist in some system-model relationship (that it maps the elements of a system onto the elements of a model). The second describes the abstraction aspect: the model does not completely reproduce the original system's properties, but reflects

a relevant selection. The word "relevant" entails the third feature: that the model be usable for some purpose in place of the system it models.

Kühne equates the abstraction function α which defines the relationship between a system S and a model M of the system with the composition of three functions: a projection π, which reduces some of the features of the system[9], an abstraction of these projected features α', and a translation τ to a representation (the modelling language). This can be written:

$$M = \alpha(S) \qquad \alpha = \tau \circ \alpha' \circ \pi$$

Depending on the intermediate abstraction α', Kühne defines two kinds of role a model can take in a system-model relationship: *token model* and *type model* (cf. Figure 3-1). A token model, also called "representational model", "instance model" or "extensional model", describes all the elements in a system (modulo projection). With token model abstraction, α' is the identity function, as no further abstraction is done than projecting the relevant properties of the system (e.g. keeping city names, road names and the links established between cities by roads, and ignoring number of lanes, position on the map, etc.) and translating this into some representational language (e.g. a UML object diagram).

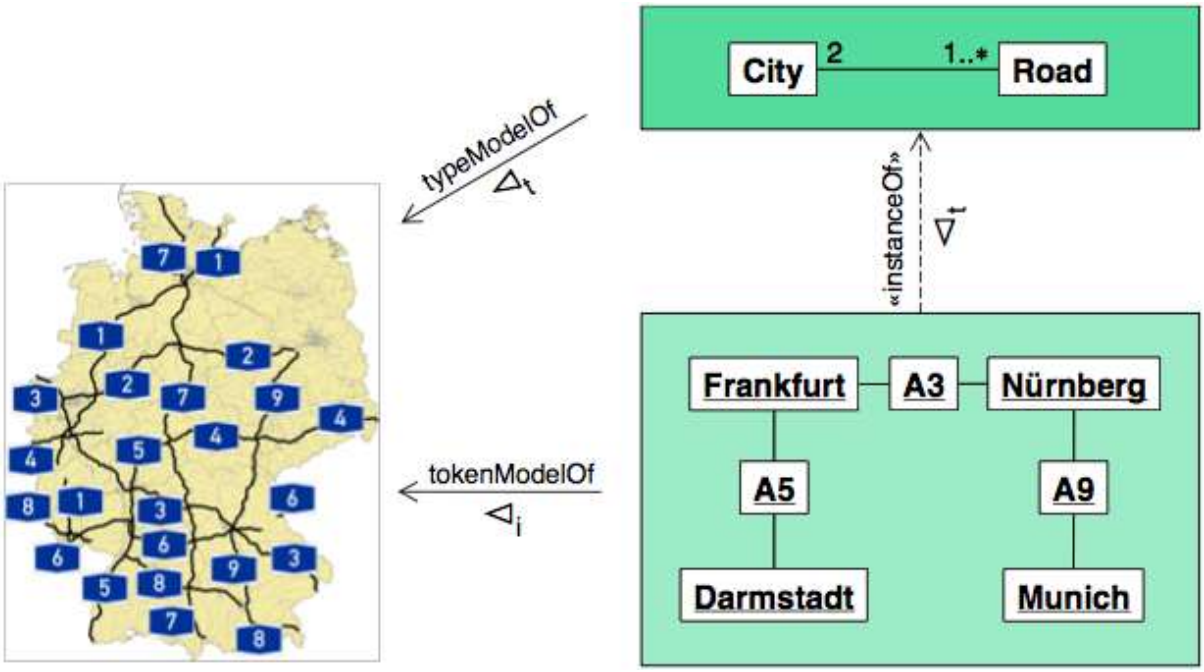


Figure 3-1 Examples of token model (denoted $\triangleleft_i$) and type model relationships (denoted $\triangleleft_t$). (Kühne, 2006, p. 5)

[9] this projection is used in the mathematical sense of reducing dimensions: a sphere projected in two dimensions can be considered a circle – this loses some, but not all information about the sphere.

A type model, also called "classification model" or "intensional model" gives a concise description of a system by using the properties of the objects in a system to classify them. With type model abstraction, α' is a classification function which assigns elements which are considered to be equivalent to each other with respect to certain properties to a single type.

Another way to describe type model relationships is to consider them in terms of extension and intension. If *C* is the principal concept associated with a type, then the intension of C is the conjunction of predicates which characterise the elements belonging to that type. For example, the intension of the concept *rectangle* is something like *is a quadrilateral* and *is a parallelogram* and *has four right angles*, etc. (the set of predicates is not necessarily finite, in particular when one or more logically imply others – it can however be useful to define a minimal set which an entity *must* fulfil in order to be of a given type). The extension of a concept is the set of all the elements that fall under that concept (e.g. all the rectangles in the world). The relationship between the two is that the extension of a concept is the set of all elements which fulfil the intension of the concept (i.e. for each predicate in the intension, this predicate is true with regard to each element in the extension). A type model is related by intension to the system it represents. Each element of the system will be mapped onto a type whose extension includes that element – or, put another way, the element will fulfil the intension of that type.

A third form of abstraction is generalisation. Whereas classification assigns a single concept to many elements, generalisation assigns a single (super-) concept to many other concepts. Kühne explains that it is only appropriate to generalise a system which is *already* a model holding some kind of type model relationship with another system. As such, the resulting generalised model will also be a type model for the initial system. Furthermore, given a concept $C_{special}$ belonging to a model and another concept $C_{general}$ which generalises the former concept and belongs to a model which generalises the former model, any element which fulfils the intension of $C_{special}$ will fulfil the intension of $C_{general}$. Put another way, the extension of $C_{special}$ is a subset of the extension of $C_{general}$.

3.2.1 What kind of model is a replayable?

Our aim in defining the notion of a replayable can be explained at two levels. First, a replayable is a concept which has a certain intension: can be visualised, can be synchronized, can be replayed, can be enriched, can be transformed to produce a new replayable. We

defined this concept in such a way that its extension *at least* covers the analytic artefacts described in the state of the art and needed by our case studies.

Second, the idea of elaborating the model for a replayable is explained in Figure 3-2. Assume a number of replayables which already exist "in the wild" (or rather in the work of researchers – potentially using certain existing tools to help create these replayables). We want to create a software system which allows each of these replayables to be represented by the system. For the system to represent these replayables, it must be able to create token models of each of them (preserving at least the most meaningful content of the data in the projection). The software system itself will be a type model for these token models, with each replayable and its components being abstracted up to generic replayable and component classes (in the case of an object-oriented programming language). What we now want to do is define a replayable model (which is not a fully implemented system) which a) will represent a certain number of concepts which establish the properties defined by the intension of a replayable and b) will be a token-model of the system. By constructing this model which is descriptive with regard to replayables, we will also have a model which is prescriptive with regard to the system we want to build.

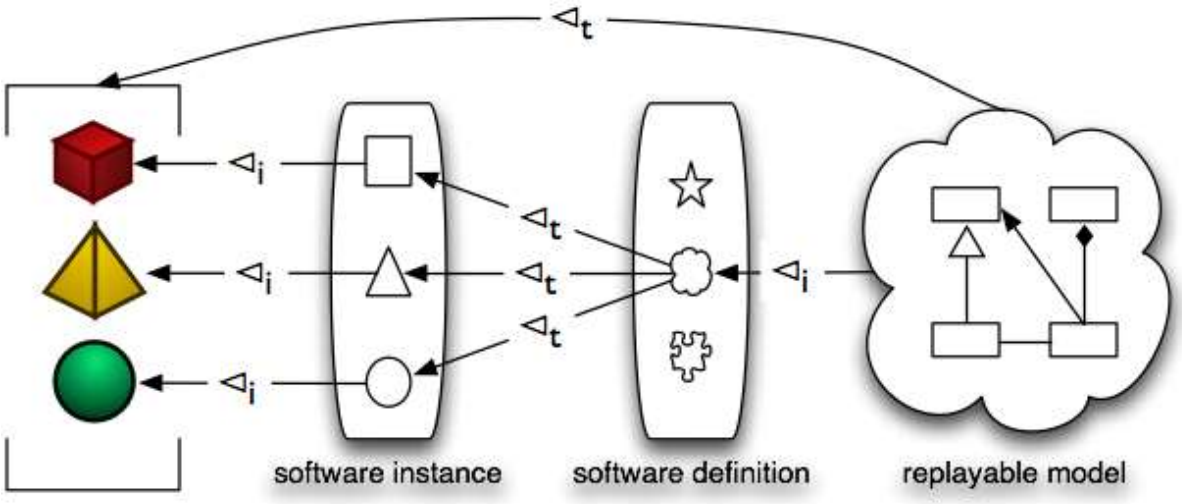


Figure 3-2 Illustration of the system-model relationships between replayables and replayable models. All replayables (left) are described (in a type-model relationship) by the replayable model (right). This replayable model is used as a token-model to describe a part of the software environment to be designed (middle right). In an instance of this software (middle left), it is possible to create different replayables which are token-models of the replayables we initially described.

This model will necessarily be a reduction with regard to the properties of the system it specifies. By examining the potential properties of replayables further, we will attempt to define those which maximise the expressive power of the model.

3.2.2 What are M-traces?

As reported in section §1.4.1.4, Settouti *et al*., 2009 formally describe a trace-based system. In order to situate our work with regard to trace-based systems, it will be instructive to examine the relation between the different parts of an M-trace (which refers to a trace, accompanied by a model), in particular to understand what kind of model the "M" in M-trace stands for (cf. Figure 3-3).

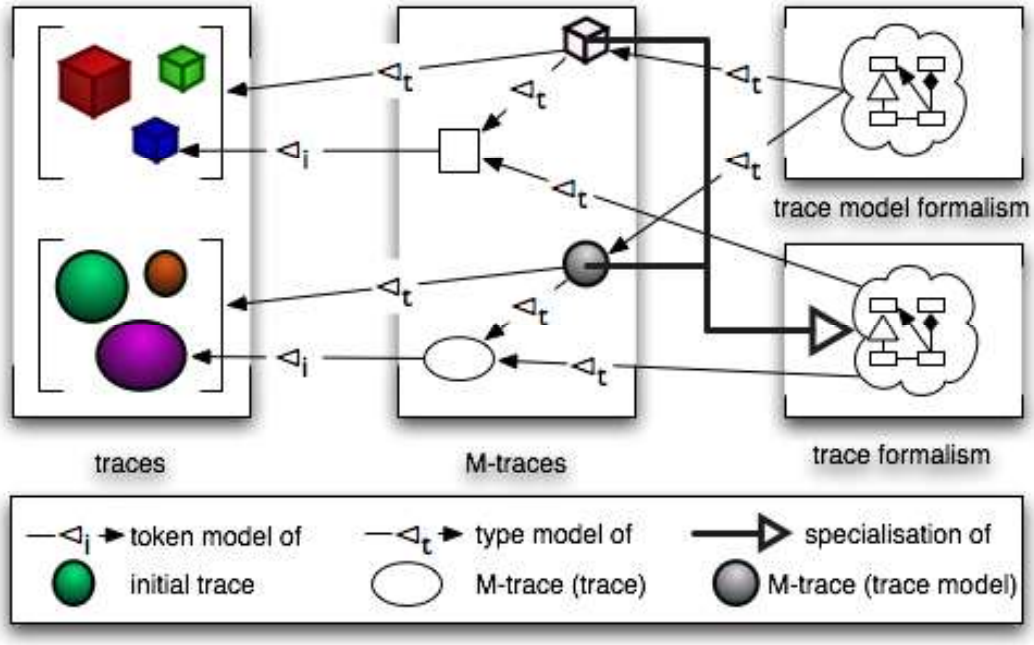


Figure 3-3 Explanation of M-traces according to the notation defined by Kühne (2006). Different kinds of "original" traces (left) can be classified as being similar (e.g. because they are produced by the same tool). Each of these "original" traces is represented by an M-trace (middle), composed of a token-model for the trace and a type model for the class the trace belongs to. A trace model formalism (right above) is used as a type model for all trace models. The trace component of M-traces (middle) are described both by the trace model (middle) and the trace formalism (right below). Trace models are a specialisation of the trace formalism.

When representing a trace in a trace based system, this trace is one of the many traces which could be collected from the tool which is being traced. A (type) model can be described for each "kind" of trace. An M-trace has two elements: the *trace* itself, which is representation (token-model, denoted $\triangleleft_i$) of the "original" trace; and the *trace model*, which is both a type-

model (denoted $\lhd_t$) for the trace and for the "original" trace. Settouti and colleagues, describe a formalism (we use the term formalism because the term "trace model" is already taken, with no special definition in mind[10]) for describing M-traces, which can be decomposed into a formalism for describing traces and a formalism for describing trace models. The trace formalism defines that a trace is made up of events and relationships. The trace model formalism in turn defines that a trace model describes the types that events and relationships can have and the constraints between them. In other words, the trace formalism is a generalisation of all the trace models which can be defined.

The use of this way of proceeding is that each M-trace will both conform to its trace model and to a general trace formalism. Some aspects of the trace-based system will assume that every trace conforms to the trace formalism (data representation, components of queries such as data access, time interval calculations, etc.). Other aspects of the trace-based system will assume that a trace conforms to its model (definition of which queries can be performed on a given trace, which specific data can be accessed in a given trace, which constraints can be applied to data when it is imported into the system to ensure its integrity, etc.).

As a final remark, we should note that our replayable model is intended to be on the same level as the M-trace formalism, *not* as the trace model of an M-trace. In the rest of this dissertation, to avoid confusion, we will use the term *trace model* to refer to the trace formalism and *specific trace model* to refer to the model which defines a particular trace or class of traces.

3.3 A model for replayables

A first distinction which can be made with regard to replayables is that between the abstract data represented in a replayable and the visible (occasionally audible) representation of this data. In the remainder of this dissertation we will call the former a *replayable* and the latter a *replayable visualisation*[11] (cf. Figure 3-4). In this view, visualisation is an operation which transforms a replayable in a such as way as to make it "viewable" by a human being – and without which a human analyst cannot be aware of the data contained in the replayable. It

[10] We could also use the word "meta-model", but Kühne (2006) imposes some very strict conditions on when the word meta-model can be used. It is not relevant to this discussion whether the M-trace formalism is a meta-model or not but we nevertheless avoid this term.

[11] Meaning that it is the visualisation of a replayable – as opposed to a visualisation which is replayable (in spite of the fact that the former may often imply the latter).

does not alter the replayable or construct a new replayable in any meaningful way. It is, however, this visualisation which can be replayed and synchronised (using the data contained in the replayable).

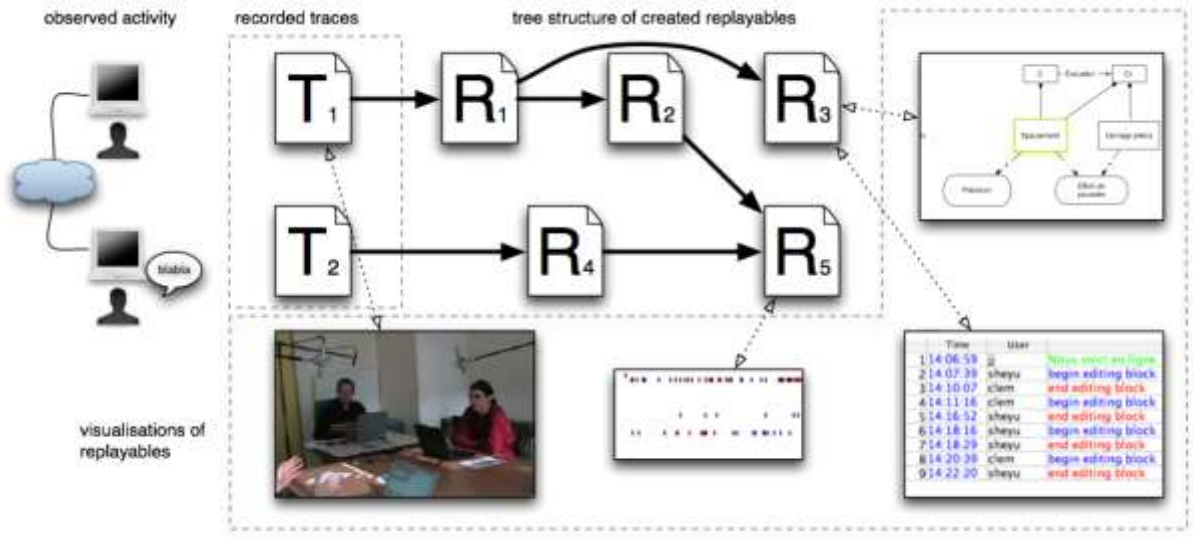


Figure 3-4 Illustration of a series of replayables created from the recorded trace of an observed activity. Each replayable can be visualised in one or several ways.

The creation of a replayable can happen through automatic transformation, manual segmentation, or the selection or reordering of the elements of an existing trace or replayable. However, as illustrated by many analysis tools (video players and tool replayers in particular), there is no obligation to *create* a replayable before constructing a visualisation which can be used to analyse an existing trace. We therefore can conclude that trace artefacts are a subset of replayable artefacts.

This leads us to ask three questions : are all replayables traces? How can we use existing trace models to build a replayable model? In what way(s) are replayables different from traces?

The answer to the first question is “no”. If we consider the definition of a trace given in §1.3, that traces are a recording of the observable events of an activity, some transformations (segmentation in particular) give rise to objects which represent an analyst’s interpretation of this data, rather than a recording. Furthermore, enrichment adds data to replayables which goes beyond the notion of a recording. While Choquet & Iksal (*op cit.*, cf. §1.4.1.2) define traces as possibly including “subjective traces” (i.e. events which are inserted/generated by a human), this does not afford the subjective enrichment of the elements of pre-existing traces. In other words, while traces are *recordings* of an activity, necessarily containing some level of subjectivity (which researchers will attempt to restricted to a minimum), replayables are

artefacts which are designed to be *proxies* for an activity, used by researchers to construct their analyses and interpretations (with whatever level of subjectivity is considered appropriate with regard to a given research tradition).

That not all replayables are traces (according to certain definitions of replayables and trace) does not prohibit us from using existing trace models to construct replayable models. Because of the relationship between traces and replayables, this seems like the best place to start, and it is possible that a trace-model could be used to model more than just traces. Indeed, the M-trace model provides nearly all the elements needed to model replayables. We could begin by tentatively modelling a replayable as being a set of timestamped *events* (sufficient to enable replay and synchronisation) and of *links* between these events. The events which compose a replayable would be described as having a set of named and typed *facets* which are used to model various aspects of an event: what happened, to whom, which tool was used, and so on. Transformation would then be defined as the creation of a replayable composed of new events and links.

This model would not address certain peculiarities of enrichment. While relationships between events can be addressed, the same is not true of codes and annotations. These enrichments could be achieved by adding new facets to events in existing replayables. For example, given a replayable composed of two events (represented in Javascript Object Notation (JSON, 2009)).

[{time: 1, speaker: "john", utterance: "are you well?"},

{time: 2, speaker: "jack", utterance: "yes, thank you"}]

this replayable could be enriched with the categories "question" and "answer":

[{time: 1, speaker: "john", utterance: "are you well?", code: "question"},

{time: 2, speaker: "jack", utterance: "yes, thank you", code: "answer"}]

But this would make it difficult to share enrichments among researchers without having to re-communicate data which has already been shared. Furthermore, suppose that in Figure 3-4 a category were added to an event in replayable *R3* and that this event also be present in replayables *T1, R1, R2* and *R5*. It seems natural that this category should be applied to all instances of this event, and should therefore appear next to this event in these other replayables. To fulfil this need, we propose: on the one hand that each event of a replayable be considered the representative of (or proxy for) an observable event (which may in turn refer to a set of events) recorded in the trace; on the other hand, enrichments should be stored

separately from replayables, forming a sort of shareable "library" which carries the responsibility, when a representative of an observed event is enriched in a given replayable, to enrich all the representatives of this event.

This model is illustrated by Figure 3-5. The *trace* becomes a special kind of *replayable* which records a number of *observed events*. These events can be enriched by *relationships* (unary in the case of codes, binary in the case of links – new kinds of enrichment can be created with other kinds of relationship cardinalities) which associate supplementary *facets* with the observed events they enrich. All events *represent* an observed event, are composed of a certain number of facets and have a timestamp. They can fetch the additional facets provided by enrichment via the observed event they represent. A set of relationships can be stored together, forming an individual *enrichment*.

By *observed event*, we do not necessarily mean that it is present as a discrete event in the trace, but that it was present in the observed activity. As such, an observed event can represent a time interval in a non-digital trace or any subset of the events in a digital trace. Among all the possible events in a collection of traces, it is up to researchers to identify which ones are pertinent to a given analysis.

Our previous example would thus be stored as:

{replayable: [{time: 1, speaker: "john", utterance: "are you well?", represents: "T1:1"},
{time: 2, speaker: "jack", utterance: "yes, thank you", represents: "T1:2"}],
enrichment: [{associate-1: "T1:1", code: "question"},
{associate-1: "T1:2", code: "answer"}]}

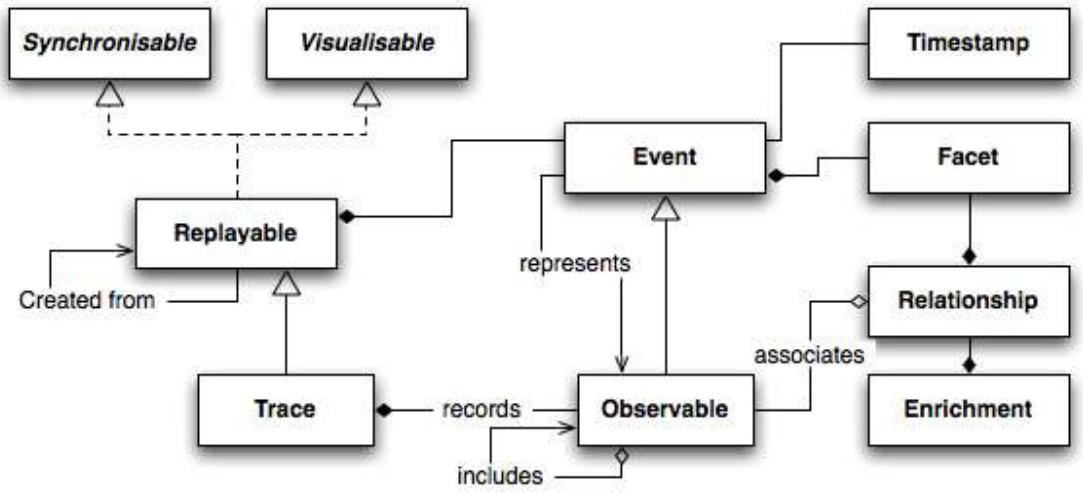


Figure 3-5 UML class diagram showing the main aspects of the proposed model for replayables.

Some elements which are pertinent to analysis do not seem to be representable as timestamped events. However, objects such as ideas, actors and groups can be thought of as existence events which begin with the first appearance of the objects and finish when they no longer exist. The relationships between objects and events related to them can then be described. For instance, the facet *speaker: "john"* could be "factorised" into an event representing John's existence as a speaker over the whole corpus and a relationship between this new event and all the events which originally had the *speaker: "john"* facet[12]. Documents which represent a final product, the assignment or other elements which are part of the analysis context are more problematic. While they could also be given "existence" events, analysis might want to refer only to parts of the document, rather the whole. Existence events could also be given to these individual parts but the exact nature of these new objects (documents and parts of document which are considered relevant) is not necessarily fully accounted for in the proposed model (as they are not primarily events). The presence of such documents in the state of the art is so rare however, that we consider the proposal of existence events to be sufficient to address them, at least within the scope of this dissertation[13].

In this model, while some relationship information may be stored in the actual *file* or *database* containing the trace, we consider that it is not part of the trace itself, but an enrichment which can be considered separately. In the case where the relationships are stored through facets of the events in the trace (e.g. an id and parent-id facet in a forum trace), we consider a useful operation to be the creation of an enrichment through *factorisation*, in order that these relationships be stored explicitly. In treatments of the trace such as transformation and visualisation, relationships can be "expanded" back in, as necessary, such that the "virtual" facets provided by enrichments are considered on the same level as the "natural" facets (e.g. when searching for events which contain the value "answer" in the code facet, there is no need to consider whether that facet comes from an enrichment or not). In cases, where a relationship is associated with several events, the "link" nature of the relationship may also be preserved (e.g. for visualising links between events). When considering a replayable, we expand in all the relationships which can be applied to events of that replayable.

[12] Although "speaker: john" is only a general truth if "john" unambiguously refers to the same person throughout the corpus.

[13] See Lalanne & Ingold (2004) whose approach for the recording and restitution of static document usage in meetings could be a first step towards a fuller integration of non-temporal documents within a temporal context.

3.3.1 Properties of replayables

Before further describing our model in relationship with other trace models, we examine the properties different replayables can have in order to determine the necessity and usefulness of defining a means for describing a *specific replayable model*, or in other words, of describing classes of similar replayables. Such a model would describe, for a given replayable type, the events that it can contain and the facets which compose it. This model could be defined from different perspectives: it might describe the events from an ontological perspective (e.g. "send message", "read message", "navigate to page"), it might also describe the facets from a similar perspective (e.g. "user", "message", "tool") or from an implementation perspective (e.g. "string", "integer", etc.). While an "ontological" model would be useful for embodying knowledge about the replayable and ensuring the runtime safety of the implementation (i.e. that only the correct kinds of queries and transformations can be applied to a given trace), the knowledge it embodies necessitates that it be too specific, only applying to certain kinds of traces and causing transformations which could be defined generically to only be applicable in a given instance. This problem can be addressed in two ways.

First, a means for describing specific trace models could be defined, including a partial ordering on models, creating a hierarchical type system whereby a given replayable belongs to its actual model (or type) and to the generalisations (or super-types) of that model. In this way, generic transformations could be defined on a general model and applied to all replayables of that type (either directly or through the ordering). This would be cumbersome for two reasons – the first is that it would be difficult (maybe impossible[14]) to provide model authors with the means to have their models be coordinated (in particular, how could an author predict that their model will be a subtype of as yet non-existent other models). The second is that as many ad hoc replayables are created during analysis, each of these would either have to be modelled by the researcher (an extra cost which would severely limit the researcher's creativity), or their model would have to be calculated (again a proposition whose difficulty we have yet to ascertain).

Second, mindful of the above problems, we could further specify the types of facets which exist and provide such a typing mechanism to trace authors. A given event's type could be calculated from the types of the facets which compose it, just as a given replayable's type

[14] Having not attempted to create such a type-system, we cannot speak further to this possibility. The difficulty in creating such systems for object-oriented programming languages (including multiple inheritance, etc.) suggest the extreme difficulty of this task. Indeed, in general, our considerations seem to follow the same perspective as the discussion of class vs. prototype languages and statically vs. dynamically typed languages

could be calculated from the events within it. Provided adequate description of transformations, the type of the result of a transformation could be automatically calculated from the type of the replayables serving as an input. The question we must then answer is whether we are able to create a list of types and at what level these types should be defined.

In order to answer this question, particularly mindful of the use of such models for analysis, we explore the properties of a replayable which determine:

- What visualisations can be created;
- The usefulness of a replayable for a given analytic need (e.g. why have a transcription rather than – or in addition to – a video?);
- What transformations can be performed;
- How generic transformations should behave (e.g. if I describe a generic fusion transformation which combines the events of two replayables into one, what is the model of the new replayable?)

Replayable properties can be described at two levels: that of the events which make up the replayable and that of the facets which describe these events.

At the event level, two kinds of *sampling* can be distinguished. *Continuous sampling* (which might also be termed "probing") occurs at regular intervals with no regard for the recorded activity. Video and audio signals are typical examples (although in the case of some audio recordings, the analog signal is recorded continuously rather than at intervals), as is mouse movement. *Discrete sampling* (or "triggering") occurs when the events are created in relationship to the activity being recorded. It can be further broken down into medium-defined sampling (an action made in a computer environment, a speech act, etc.) and researcher-defined sampling, adapted to the kind of analysis being performed (e.g. a step, an activity, etc.). The changes in sampling type depend on the adequate granularity for analysis: the sampling must be such that the individual events somehow exhibit the aspect of the activity which is under study. For example, audio (continuous sampling) might be necessary to observe the precise overlap between two speakers if this overlap were relevant to the analysis. A transcription would, however, be sufficient if only the lexical content were of importance.

At the facet level we can distinguish different forms of *representation*: *digital*, *textual*, *symbolic*, *identifying* and *scalar*. These forms belong in a type hierarchy (cf. Figure 3-6). All facets have *digital representation* which require a machine to read and understand them: we can cite examples such as video (a replayable whose events each contain a single image

facet), a digital trace which, given a replayer, can reproduce what was seen on screen, or an audio signal. Some forms of digital representation are so common that "universal" machines exist to understand them. One such form is *textual representation*, which is directly readable by humans and can be indexed (e.g. for information retrieval purposes). The value of a facet is the string which represents the text. *Symbolic representation* is a special form of textual representation which uses a controlled vocabulary on top of which an additional semantics can be added. The value of each facet is a set containing a token (or string) for each symbolic value. Such a representation affords the ability to create statistics such as the number of times a given word of the vocabulary is used. Identifying representations, in turn, are a special form of symbolic representation where each word in the vocabulary matches an object with a persistent identity which is relevant to analysis. Such objects might be usefully "factorised" out as existence events. The other kind of digital representation with a well known machine are *scalar* representations which represent numbers in some given unit. Scalars and expressions on scalars can be used in queries using equality and comparison operators.

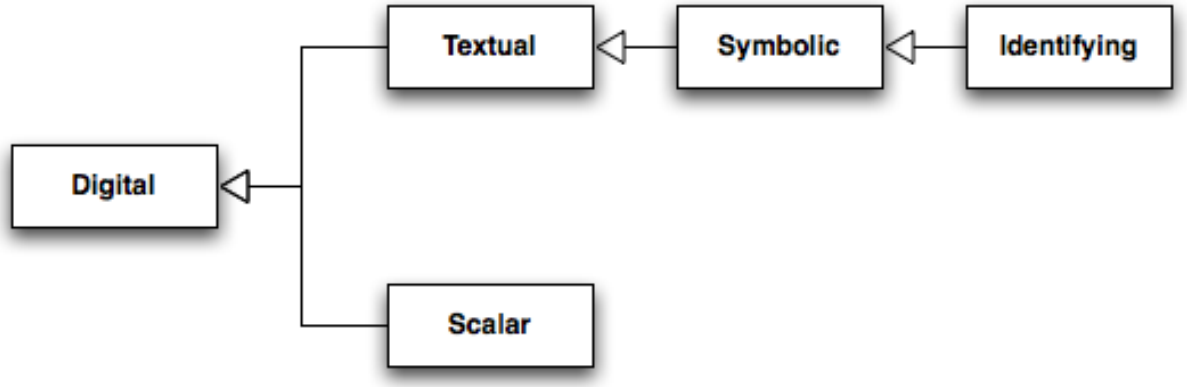


Figure 3-6 Hierarchy of facet representation forms

A final distinction can be made between facets which are directly provided by the trace, those which are calculated from the trace and those which are added by the analyst. The provenance of a facet will inform how "subjective" it is and allow the analyst to take this into account when using it, potentially having to validate it (e.g. by examining the source data more closely or having a second person independently perform a given analysis step).

While the first and last properties (sampling and subjectivity) can highlight the necessity of a step, or on the contrary, its superfluous cost, we claim that they are not otherwise useful when it comes to exploiting specific models of replayables – i.e. software support for analysis within which each replayable has a model would not benefit from these properties being present in the model (beyond making these properties known to the user).

The representation, however is of capital importance for defining visualisations and transformations, which we will address in sections 3.4 and 3.5. It is notable that in this description, the model does not know about such concepts as users, tools, messages and so on. We choose this for two reasons. The first is that defining an exhaustive list of such concepts may prove to be impossible (cf. §1.4.1.1 and the common format defined by the Cavicola group). The second is that, aside from analyses which are too specialised for consideration within the scope of this dissertation and which are designed with such concepts in mind (e.g. examining user participation, an activity which is particular to certain kinds of analyses), we have not actually found any benefit to be gained in analysis which cannot be done by naming a facet "user" and letting researchers use this information to interpret the data in their analyses.

None of the benefits we laid out in defining specific replayable models seem to justify going to the trouble of formally defining a type system for replayables, but this may be an interesting theme to be addressed by future research. For the purposes of this work, we use a shorthand which allows us to describe the type of a replayable through the types present in its events. We can describe these events by bags[15] of the representation type of their facets. We name the types in Figure 3-6: $R_{digital}$, R_{text}, R_{symb}, R_{ident} and R_{scalar}. An event containing a textual facet, a digital facet and two symbolic facets would be noted: { R_{textl}, $R_{digital}$, R_{text}, R_{symb}, R_{symb} }. A replayable's type can be described as the greatest common sub-bag of the bags of each of its event.

3.3.2 Modelling replayables with M-traces

In order to provide means for future work to explore the notion of replayables more formally and as a means of illustrating the benefits of the proposed model, we attempt to describe this model in terms of the formalism proposed by Settouti *et al.* (*op. cit.*; cf. Figure 3-7). Specifically, we highlight where incompatibilities exist.

[15] Although we use a shorthand of set notation, we intend them to be *bags* in which duplicate elements are possible, as events having multiple facets of the same type are perfectly legal.

Definition 4 ($\mathcal{M}$-Trace). *A $\mathcal{M}$-Trace is a tuple*

$$\mathbb{T}^r = (\mathcal{M}_{\mathbb{T}^r}, \mathcal{E}_T, O, id, \lambda_C, \lambda_R, \lambda_A, \lambda_T)$$

consisting of

- *a trace model* $\mathcal{M}_{\mathbb{T}^r} = (\mathcal{T}, C, R, A, \text{dom}_R, \text{range}_R, \text{dom}_A, \text{range}_A)$,
- *a temporal extension* $\mathcal{E}_T$,
- *a finite set* O *of observed elements, disjoint from* C, R *and* A
- *an invertible total*[2] *function* $id : O \to \mathbf{V}$ *(i.e. id is bijective between* O *and* $\text{range}(id)$*),*
- *a total function* $\lambda_C : O \to C$ *called element type labeling,*
- *a relation* $\lambda_R \subseteq O \times O \times R$ *called relation type labeling,*
- *a partial*[3] *function* $\lambda_A : O \times A \to \mathbf{V}$ *called attribute labeling,*
- *a total function* $\lambda_T : O \to \mathbf{I}(\mathcal{E}_T)$ *called time labeling.*

Furthermore, a trace must be consistent *to its model,* i.e. *verify the following constraints:*

- $\mathcal{E}_T \in \mathbf{I}(\mathcal{T})$
- $\forall (o_1, o_2, r) \in \lambda_R,\ \lambda_C(o1) \leq_C \text{dom}_R(r) \wedge \lambda_C(o2) \leq_C \text{range}_R(r)$
- $\forall (o, a, v) \in \lambda_A,\ \lambda_C(o) \leq_C \text{dom}_A(a) \wedge v \in \text{range}_A(a)$

As a convenient notation, we will sometimes use λ_R as a function such that $\lambda_R(o_1, o_2) = \{r, (o_1, o_2, r) \in \lambda_R\}$.

Figure 3-7 Definition of an *M*-trace, as proposed by Settouti *et al.* (2009), pp. 7-8

Using the definition of an *M*-trace, all replayables would conform to a model (cf. Figure 3-8) and contain a set of typed observable elements, which are located in time, each of which having a unique identifier and a set of attributes.

Definition 3 (Trace Model). *A Trace Model is defined as a tuple*

$$\mathcal{M}_{\mathbb{T}^v} = (\mathcal{T}, C, R, A, \mathrm{dom}_R, \mathrm{range}_R, \mathrm{dom}_A, \mathrm{range}_A)$$

consisting of

- *a temporal domain* $\mathcal{T}$,
- *a finite set* C *of observed element types (or classes), with a partial order* $\leq_C$ *defined on it,*
- *a finite set* R *of relation types, disjoint from* C*, with a partial order* $\leq_R$ *defined on it,*
- *a finite set* A *of attributes, disjoint from* C *and* R,
- *two functions* $\mathrm{dom}_R : R \to C$ *and* $\mathrm{range}_R : R \to C$ *defining the domain range of relation types,*
- *two functions* $\mathrm{dom}_A : A \to C$ *and* $\mathrm{range}_A : A \to \mathbf{D}$ *defining the domain and range of attributes,*

It must also hold for any two relations r_1 *and* r_2 *that*

$$r_1 \leq_R r_2 \Rightarrow \mathrm{dom}_R(r_1) \leq_C \mathrm{dom}_R(r_2) \wedge \mathrm{range}_R(r_1) \leq_C \mathrm{range}_R(r_2)$$

Figure 3-8 Definition of a Trace model, as proposed by Settouti *et al.* (2009), p. 6

Events can be denoted as being representatives of other events through a special representation relationship, relating an event to the event (or events) it represents. In our model, relationships can be unary and can carry attributes. Because of this, they would have to be represented in a similar way to events (i.e. they are also observable elements but relax the constraint on the presence of a timestamp). Because in our model, the relationships between events are not constrained to events within a single replayable, this would force us to adopt a situation where all replayables within a given context (a corpus) are represented by a single trace. It would then simply be necessary to add an new kind of observable element (the replayable) which is associated with the events which belong to it. Although it appears possible to also use this mechanism to eliminate the necessity of representation (since a given event could appear in several replayables and the enrichment of that event would only need to be performed once), this would not work: representatives of the same event in different replayables can have different sets of facets.

In this form, the proposed model is also quite similar to that proposed for structuring data in the Nite XML Toolkit (cf. §1.4.3). The major differences lie in the separation into

replayables composed of events, as opposed to annotation layers (composed of annotations), and the ability to enrich events in a way which can be propagated across replayables.

3.4 Replayable visualisations

Visualisations are important because they are what enable researchers to understand replayables. Replayable visualisations can be distinguished by their means of representing time and their ability to treat the various kinds of replayables they represent. The properties of visualisations will contribute to their usefulness in analysis. In this section, we revisit the various kinds of replayable visualisations which were presented in §2.5.2 in light of the model we have defined in order to relate the properties of replayables back to analytic needs.

When time is not represented in a visualisation, replay must be used to re-establish the temporal aspect. This is the case for audio, video, visualisations in a trace replayer and, more generally, visualisations where spatial proximity is not indicative of temporal proximity (e.g. a concept map would have to be replayed in order to see the ordered appearance of elements). Replayables which are made up of events having facets in a digital representation are the most problematic in this respect: the machine enabling their visualisation most often produces a two-dimensional image, making it impossible to represent time in a spatial fashion. Analysis time is therefore increased by the necessity of replaying in order to restitute the temporal context. Furthermore, it will be difficult to restitute multiple facets in the same visualisation. As a general rule, there will be a trade off between the cost of creating a new visualisation (possibly of a new underlying replayable) which can be understood without being replayed and the cost of constantly replaying the original replayable (this partially explains the desirability of transcription and the advantage of having a digital trace as opposed to screen capture).

Tables, graphical timelines, curves, and text represent time on a spatial axis. They can also "simulate" replay and assist synchronisation by successively highlighting the unfolding of time along that axis (e.g. by selection graphical elements or lines in a table). Tables have certain restrictions with regard to the presentation of overlapping events, which can be shown on different lines in a graphical timeline, a curve or a transcription. When events do not overlap, however, or when that overlap is not relevant to a given analysis, tables have the advantage of being able to represent a large number of facets (one per column), and to be able to easily show facets which have scalar or textual representations.

Graphical timelines are well adapted to the succinct presentation of symbolic facets, as each word of a vocabulary can be assigned to a graphical property such as form, colour or vertical positioning (assuming horizontal timeline and positioning on labelled rows). The limited number of graphical features (shape, size, colour, position) and the limited range of these features (number of distinguishable colours, number of available shapes) limits both the number of facets of a given event which can be shown and the size of the vocabulary used by these facets. The extension of the vocabulary must therefore be known before defining a visualisation, limiting our ability, within this framework, of defining generic visualisations. Because of the unlimited vocabulary of textual, non-symbolic facets, it does not make sense to represent them in a graphical timeline, except by placing the text next to the graphical symbol for the event. Graphical timelines can also present (within reasonable limits) scalar facets with size and vertical position (assuming horizontal timeline and a graduated vertical axis), although such facets are best presented through curves (due to the often continuous nature of scalar data). Finally they are the most appropriate way of presenting relationships between events, unlike tables which, even in the case of treetables, can only show acyclical graphs.

Thus, the choice of the nature of a visualisation is intrinsically related to the type of the replayable and to the analytic need. Each kind of visualisation will have a limitation on the amount of data it can show and the kinds of data it is adept at showing, frequently leading to simultaneous visualisation of the same replayable in multiple ways (e.g. a graphical timeline to show relationships and help identify patterns in a coding combined with a tabular representation to show the complete set of facets of each event). The creation of interactive visualisations will also contribute to solving this problem as information can be made to appear on demand.

3.5 Transformation of replayables

The properties of replayables and visualisations we have explored help to define the kinds of transformations which can be applied to a given replayable and the motivations which might exist for a given transformation to be applied.

A transformation can be described as a function whose domain is the set of replayables which can be transformed by it and whose range is the set of replayables it produces. This description serves to understand and illustrate what properties certain kinds of transformations

confer to their resulting replayables. We describe the domain as the minimal sub-bag of facets which must be present in order to apply the transformation and the range as the minimal sub-bag of facets which are present in the result. The semantics of a transformation T: A -> B imply that the replayable must at least have facets of the types described by bag A and will produce a replayable with facets of the types described by bag B. In the case where a transformation preserves unspecified facets (because it integrally reproduces all the facets found in the source replayable), we note T: A ∪ S -> B ∪ S, to indicate that all facets in bag S (i.e. those not explicitly present in bag A) are still present in the resulting replayable.

Transcription transforms a replayable with continuous sampling and having facets with digital representations (typically audio and video), which necessitate a machine to understand them, into a replayable with discrete sampling whose events have an identifying facet (the speaker) and a textual facet (the speech act). This transformation can be noted $\{R_{digital}\}$ -> $\{R_{ident}, R_{text}\}$. The large cost of transcription can then be assessed with regard to the time saved through the resulting replayable which has a greater number of facets to visualise and perform calculations on, does not need replay to assist the visualisation, and is searchable. Depending on the most desirable properties of the new replayable, transformation can be broken up into its component steps of *segmentation* (continuous sampling transformed to discrete sampling with no preservation of event types), *textual transcription* ($\{R_{digital}\}$ -> $\{R_{text}\}$) and *identification* (S -> $\{R_{ident}\} \cup S$) of speakers. Some of these steps might be done only partially or not at all.

Filtering ($\{R_{symb}\} \cup S$ -> $\{R_{symb}\} \cup S$) is a transformation from a replayable having at least one symbolic facet into a replayable containing only the events for which this facet meets a given requirement (e.g. being equal to or different from a certain value). In order to be in a position to perform such a transformation, it can be useful to use an *identification*, or a *categorisation* (S -> $\{R_{symb}\} \cup S$) which respectively add identifying and symbolic facets. A *search* ($\{R_{text}\} \cup S$ -> $\{R_{text}\} \cup S$) is similar to filtering and can either be performed by applying a similarity metric between each facet value and a request, or by first *tokenising* ($\{R_{text}\} \cup S$ -> $\{R_{text}, R_{symb}\} \cup S$) the replayable, calculating a set of symbols (in this case, the morphological root of each of the words) from each textual facet, and then *filtering*.

The manual creation of a collection is also a form of filtering. Such an transformation could be considered as an implicit act of *categorisation*. This category could then be added explicitly to each event in the resulting replayable. A different kind of filtering is *time slicing* (S -> S), which restricts a replayable to the event within a given time period.

At any point, an enrichment can be created by factorisation (as explained in §3.3.1) of one of the facets. Expanding this enrichment back down into another replayable shows the semantics of the *folding* transformation ((A,B) -> A ∪ B), which takes two replayables whose events both represent the same observed events and, for each observed event, folds in the facets of the representative events from each replayable. This operation could be used to perform textual transcription and speaker identification separately and then combine the results. Another operation which can be performed on two replayables is *merging*. This produces a replayable which includes the events from both of the original replayables. These events are then available in the same replayable and can be included without needing synchronisation. In a merging operation, it can be useful to ensure the coherence of facets of the different event types. Sometimes facets with the same (or similar) semantics can carry different names (e.g. "speaker" and "user") and should be unified. Facets which end up with the same name should be in the same units (in the case of scalars) or use the same vocabulary (in the case of symbolic and identifying representations). Sometimes a mapping exists from one vocabulary to the other, e.g. in the case where a transcription using the anonymised names A and B to refer to the participants is merged with a digital trace where non-anonymised pseudonyms are used. This suggests the existence of two other kinds of transformation: *facet-name unification*, and *facet-value unification* which respectively map facet names and facet values in a many-to-one relationship. This is one of the rare cases where the semantics *are* important and it would be useful to assist the user in explicitly managing them.

Finally, *grouping* changes the sampling level of a replayable, combining several events from the source replayable into a single event in the target replayable. This new event is the representative of a new observed event which constitutes the set of observed events represented by the events that it combined. Its temporal extension can be considered as the smallest temporal extension which includes the temporal extensions of each of the composing events. The values of the preserved facets of this new event (i.e. those that are not provided by the grouping transformation itself) can be determined using the semantics of factorisation and expansion. This raises the question of how (and whether) multiple values can be expanded into the same facet. In the case of symbolic facets, each value is a set and the result is generally the union of these sets (e.g. the grouping of an action performed by A and an action performed by B is an action performed by the group {A, B}; similarly, the grouping of two actions performed by A is an action performed by A). In the case of textual facets, concatenation often provides a reasonable result. In the case of scalars, and digital

representations in general (this might also apply to symbolic representations in certain cases), some operation must be defined which preserves the semantics of facet in question: a scalar which represents a number of words would be combined by addition, whereas a scalar which represents a rating would be combined by arithmetic mean; many other operations could be imagined. In the absence of such semantics, a lightweight version of grouping is *wrapping*, which creates events with no facets (but with a correct timestamp) which point to a new observed event which encapsulates the list of wrapped events.

If grouping is to have a generic definition allowing the preservation and propagation of a maximum amount of information, it therefore seems necessary to include for each facet in a replayable, not only its type, but also the operation which describes how (and whether) it can behave under fusion. With appropriate semantics on a controlled vocabulary (e.g. a partial ordering representing an ontology), this could allow {UserA, UserB} to be automatically evaluated as {Group3}, and many other automated inferences, allowing the maximum benefits to be drawn from analysis of multi-scale phenomena. These considerations along with the others regarding semantics seem to indicate that future work should be inspired by the notion of *frames* (Minsky, 1974), of which our events with facets are reminiscent. More generally, the requirements of analysis seem to indicate that languages with flexible latent semantics (e.g. prototype-based dynamically-typed languages) are best suited to modelling replayables.

3.6 Conclusion on constructing a replayable model

In this chapter, we explored the reasons for defining a model for replayables in order to understand the kind of model we wished to construct, the benefits of constructing such a model and how to relate this model to existing models. We first explored the notion of a model, determining that the construction of a type model for replayables would allow us to specify a system within which token models of replayables could exist. We further determined that while a specific type model for certain classes of replayables would parallel the notion of a model for certain classes of *M*-trace (which adds semantics and determines the well-formedness of queries and transformations), using such models would not allow us to define generic transformations. For this reason, among others we chose to present a model

which does not present a great degree of formalism. Indeed, at this point in our understanding of analysis (through replayables or otherwise), we do not know enough about transformations, visualisations and kinds of analytic artefacts to feel confident in proposing a more precise formalism which would, we fear, lead us to define replayable operations in a way which restrict researchers where they should be given liberty and grant additional freedoms which serve no purpose or might event prove a hindrance.

We then defined a model for representing replayables and their enrichments. In this model, traces are a special kind of replayable, which are composed of observed events to which enrichments can be attached. Replayables contain events which have a set of facets and *represent* an observed event in the trace. By this mechanism, events can inherit the virtual facets provided by enrichment on the observed event they represent.

Seeking to examine the properties of replayables in order to distinguish those which had similar semantics, we defined that a replayable's events can be sampled in a continuous or discrete fashion. Furthermore, the facets of an event can have different kinds of representations (digital, scalar, textual, symbolic and identifying) and different degrees of subjectivity. We chose not to propose a means for describing specific replayable models, as it is difficult to balance between flexibility, analytic power and precise models. Because each replayable created during an analysis might potentially have a different model, we do not want the analyst to have to define this model each time. Instead, we let the names of facets speak for themselves, allowing analysts to use their knowledge of the corpus to understand what data is carried in a given facet.

Finally, we explored how different replayable types could best exploit the different kinds of visualisation and what kinds of generic transformations could be defined on replayables, such as transcription, segmentation, identification, categorisation, filtering, tokenisation, search, folding, merging and grouping.

4 Implementation of an environment for manipulating replayables

4.1 Introduction

In this chapter, we explain how we implemented the replayable model proposed in the previous chapter in order to provide analysts with an environment in which replayables can be manipulated. We called this environment Tatiana[16] (Trace Analysis Tool for Interaction Analysts). We first list the general requirements set out for this environment, then describe the general architecture and the specifics related to implementing the replayable data model, transformation, visualisation, synchronisation and enrichment.

4.2 General requirements

In implementing this environment, we had four specific requirements in mind:

- The environment should implement the replayable model and the operations defined on replayables. This ensures that the replayable model presented in chapter 3 is actually implementable.
- The environment should afford analysis as laid out in chapter 2. This allows the environment to be tested on real-world analysis scenarios (cf. §5.2) in order to verify that various kinds of analyses can be carried out with replayables.
- The environment should be usable by users having no specific programming knowledge.
- The environment should present various degrees of extensibility or tailorability, both from a user's point of view (configuration and composition of existing functionalities) and from a developer point of view (creation of new functionalities which integrate with the existing ones).

[16] Tatiana can be downloaded at http://code.google.com/p/tatiana

Further requirements include that it be easy to install on many kinds of machines and that the code be open source (and therefore only use libraries which are compatible with this goal).

4.3 General architecture

Tatiana is implemented using the Eclipse (Eclipse, N.D.) Rich Client Platform. Eclipse itself is an Integrated Development Environment (IDE) which is based on this platform. Eclipse manages projects through a specific file structure and provided means for the component files (source code, etc.) to be edited. The Eclipse Rich Client Platform's architecture is based on a core which loads plugins. These plugins can provide points at which they can be extended by new plugins. Tatiana's core functionality is a plugin which extends the "application" extension point. The parallel is quite simple: rather than be an integrated *development* environment, Tatiana is an integrated *analysis* environment, using the same plugins as the Eclipse IDE to manage a file structure in which replayables and their enrichments are saved and providing the means to create, view and edit them in a seamless, integrated fashion.

The general architecture of Tatiana is described in Figure 4-1 (a more extensive description of the software architecture can be found in Appendix I). The four main components are replayable creation and transformation (shown in pink), visualisation (shown in orange) and replay (shown in blue), synchronisation (green) and enrichment (grey).

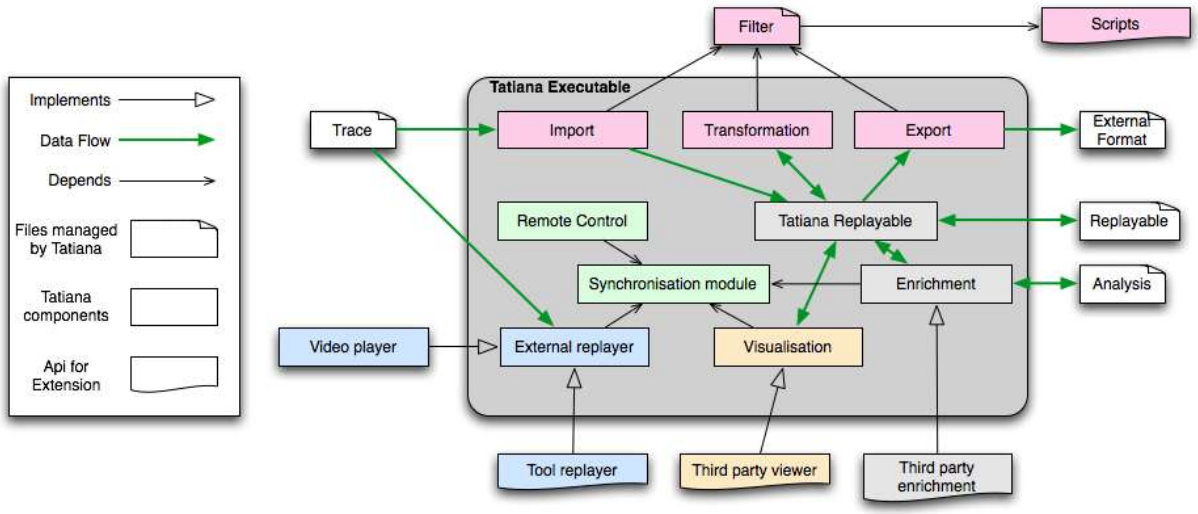


Figure 4-1 General architecture of Tatiana showing the dependencies between components and the components which are designed with extensibility in mind.

Replayables which are traces can be visualised in external replayers (e.g. tool replayers and video replayers) and can be imported to create replayables in Tatiana's data format. Such

replayables can be transformed to create new replayables and can be exported. Import, transformation and export all make use of *filters* which in turn make use of *scripts*. The latter must be written by programmers, whereas filters which combine existing scripts in new ways can be constructed by users. Replayables can be enriched in various ways according to an extensible enrichment API (Application Programming Interface). Replayables can be visualised in various ways according to an extensible visualisation API for which we provide two examples. Multiple visualisations, enrichments and external replayers can be synchronised via a synchronisation architecture. We also provide an API for implementing external replayers.

4.3.1 Tatiana data storage

Tatiana stores all its data in XML files, using an abstract data format which is similar to the tuplebase used by Knowledge Forum and KSV (cf. §1.4.2.3), or to database formats such as CouchDB (Anderson, Slater, & Lehnardt, 2009) and BigTable (Chang *et al*., 2006). These formats are all capable of storing a number of entities, with each entity having a number of named and typed fields. While each entity can have a "base" type which describes the entity intensionally (has field A: String, has field B: Integer, etc.), entities can have any number of other fields. The extension of fields of a given entity can be obtained by introspection at run-time (determining type information using principles similar to duck-typing (Duck Typing, 2009). We choose such a data format for its flexibility in defining a wide range of documents which can be parsed by a single parser and which does not require new document definitions to be written every time a new ad hoc entity type is created.

The choice of implementation in XML is motivated by the fact that many trace files are already in XML and the document nature of replayables makes it natural to store and share them as XML documents. The Tatiana concrete XML format (called Tatiana info format) is an XML tree which contains a sequence of *items* (entities), each item having a set of *infos*. Each info is named and typed. Types include time, text, HTML, number, float, anchor, sequence and boolean. Time is specified as being a time interval with a start date and a duration, which can be one of the types defined in XML for dates, times and durations, or a number of milliseconds (unix time stamp). Anchors are XML types which are used to designate a subsection of a document; they are separated into two components: a document URL and a path within that document (e.g. using XPath for XML documents). Sequences are lists which can contain any number of other values – if the order of the list is ignored and no

duplicate values are present, it can be considered as a set. If a duplicate info name is present, this is considered equivalent to the values being placed in a sequence.

All Tatiana's files exist in a single file hierarchy (an Eclipse workspace) which contains a set of corpora (Eclipse projects), each of which contains three folders: one for traces (original trace documents in their original format), one for replayables and one for enrichment files (called analyses in Tatiana). In this sense, a corpus means nothing more precise than "a set of related traces and the replayables and enrichments constructed on those traces". Furthermore, replayables and enrichments are individual files which can be shared with other researchers who use Tatiana. A special Eclipse project (named "Tatiana") contains various generic files such as those used for transformations and other non-corpus specific information.

Replayables are stored in the *Tatiana display format* (called "display" for historical reasons). This is a file in the Tatiana info format in which each item is an event and has infos called *src-anchor* and *time*. The value of the *src-anchor* info must be typed as an anchor or a set of anchors. It serves to indicate what observed event the replayable event represents (cf. §3.3). The document URL in such anchors must be a relative path within the corpus to which the replayable belongs (for reasons of transferability between computers). The value of the time info must be a typed as a time. The other infos represent the facets of the event. In order to make Tatiana slightly less confusing, all infos are called facets, including those named time and src-anchor. The order of the events in the sequence is important and is not necessarily chronological (for example, a blog might best be represented by posts with the corresponding comments placed immediately after each post rather than chronologically). Sequence facets and multiple facets with the same name are not yet handled by Tatiana (with exception for the src-anchor facet).

An API is provided for accessing the data contained in replayables (cf. Appendix II which also provides a DTD for display files). This API transparently integrates the facets provided by enrichments. Enrichments and other kinds of data are also stored in the Tatiana info format and will be detailed in later sections.

4.4 Transformations

As traces are also replayables, a replayable manipulation environment must be able to handle two kinds of files: the replayables that it creates and the traces which constitute the initial

corpus (only the former are called replayables in Tatiana). In order to do this, we make two assumptions:

- Files are either in some XML format or in a well-known multimedia format;
- In the case of XML files, Tatiana does not need to know about the semantics of the data it treats aside from a means for converting the trace into a well-formed Tatiana display file.

In Tatiana, transformations (the creation of new replayables) can be performed automatically or manually (but these two mechanisms cannot be combined for a single replayable, cf. below). Manually created replayables (or *editable replayables*) are initially empty. New events are created by selecting one or more events in another replayable or selecting an interval on the remote control (for the segmentation of multimedia files) and then using the "create new event in current replayable" function. These events can also be reordered and deleted.

Tatiana's automated transformation mechanism can be applied indifferently to XML trace files and to replayables in the Tatiana format (it is left up to the analyst to ensure that a given transformation is both performable and meaningful). In Tatiana, new replayables are created by using *filters*, which use a format particular to Tatiana to describe a workflow through which inputs are transformed by *scripts*. Filters provide information to the user about what kind of inputs are expected: trace files, replayables, enrichments, facet names, numbers, times or strings. They then "route" that data to scripts, written in XQuery (W3C, 2007). Filters which accept trace files should incorporate a script which is specifically adapted to a given trace format. As such, allowing Tatiana to handle a new kind of XML trace is as simple as creating an appropriate script to "import" it. The transformation mechanism is not limited to replayable creation and transformation, but can also be used to export to other formats, such as Excel or HTML.

Carletta Kilgour, O'Donnell, Evert, & Voormann (2003) note that XQuery cannot easily be authored by non-technical users. However, Georgeon (2008) observed that when presented with an existing query (written in SPARQL, a language for querying RDF) non-technical users were frequently able to modify it in order to adapt it to a different need. In Tatiana, filters are intended both as a means for reusing components and as a means for non-technical users to combine generic scripts in order to create transformations which fulfil a specific need.

For example, in our own analysis work, we needed to delimit periods of activity and inactivity in a trace file. We could have written a specific transformation for that trace format which imported it and added activity delimitation events. Instead, we created three re-usable component scripts and combined them in a single filter (cf. Figure 4-2). The first imports data from the trace file parameter. The second takes as parameters a replayable and number which defines the minimum duration for a period of inactivity and creates a replayable with begin-activity and end-activity events. The third merges two replayables to create a single replayable which incorporates the events from the other two. The first script can now be used in any transformation which must import that kind of trace. The second and third can be used on their own to transform any kind of replayable. A graphical editor for filters has been implemented but is yet to be incorporated into Tatiana. Filters are stored in a subset of the Tatiana info format (c.f. Appendix V).

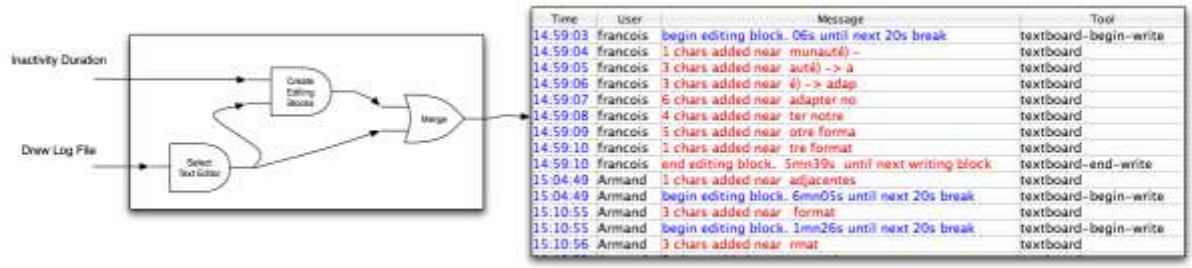


Figure 4-2 A Tatiana filter which combines three component scripts to add information about activity and inactivity. In a visualisation of a sample replayable created by this filter, the events originally present in the trace file are in red and the inserted delimiters are in blue.

Editable replayables are integrally stored on file. Replayables which are created automatically are stored as the parts of the transformation used to create them (i.e. the filter to be used and the values of the different parameters passed to the filter). The results are cached to avoid filters being rerun needlessly. This mechanism ensures that if the source of the transformation changes (e.g. it was a manually created replayable to which an event was added), the result is re-evaluated and changes also. It also allows the parameters of transformations to be easily modified. For this reason, replayables which are created automatically cannot be edited (i.e. events cannot be added, deleted or re-ordered): each edit action would have to be stored and applied sequentially each time the transformation was re-evaluated, creating the difficulties of storing such edits in an unambiguous way (the resulting replayable might drastically change if transformation parameters were changed) and applying them without too much time being needed (the time taken would be proportional to the number of edits performed, which could

be numerous over the lifecycle of a replayable). To circumvent this restriction, it is possible to explicitly save automatically created replayables as editable replayables. They are then no longer associated with the transformation used to create them, but can be edited. Editable replayables, just like automatically created replayables, can serve as input for scripts.

Scripts can be added to Tatiana simply by placing them in the scripts folder. There is currently no type safety implemented on transformations. Authors of scripts carry the responsibility of producing a file which is valid according to the Tatiana display format. They should also ensure that, for each event, that event's src-anchor points to the event it represents (i.e. create a new unique anchor in the case of import, propagate the existing anchor in the case of a transformation which does not change granularity, and produce a set of anchors in the case of a grouping). If a transformation cannot be executed or its result cannot be read for any reason Tatiana simply produces an error message. Authors should also provide a filter which documents the expected inputs for every script (cf. Appendix V). We provide a set of sample scripts which:

- filter the events by facet value (e.g. only those where the "user" facet is "john")
- search for a word and return the events where it can be found
- merge two replayables
- limit a replayable to a given time interval
- find all the occurrences of a sequential pattern
- import from Excel, atom (blogs), DREW, Elan, CoFFEE, Knowledge Forum

The transformation mechanism is also suited for export and aggregations, in spite of the fact they don't create Tatiana replayables. Such "non-replayables" can be visualised in the tabular view (provided they are written in the Tatiana format) and with any other visualisation plugin which does not absolutely require the presence of the src-anchor and time facets (i.e. the graphical timeline could not show them). They cannot, however, be synchronised, replayed or enriched. Examples include scripts which:

- detect frequent sequential patterns (for example Paul answers John who answered Peter: 25 times, Paul answers Peter who is answering John: 30 times, etc.)
- calculate the contingency table between the values of two facets (e.g. the distribution of intervention type by user)
- count the words in a given facet of each event and calculate their distribution by user, coding category, group, etc.

- export to excel, html etc.

No other transformation mechanisms currently exist for Tatiana but it might be feasible to accept the result of database queries as input for transformations and to "outsource" transformations to other services. Although XQuery is Turing-complete[17], some kinds of transformations might be better expressed in different languages such as Java. This can currently be "simulated" by creating a visualisation plugin which performs the transformation, as detailed below.

From a user's point of view, only filters are accessible (which in turn transparently call scripts). A list of filters to choose from is presented when the user activates the "create replayable from filter" function.

4.5 Visualisation

Just as Tatiana distinguishes traces and Tatiana replayables, the visualisation mechanism of each is slightly different. Traces must be visualised in external replayers which use the synchronisation API (cf. §4.7) but cannot benefit from enrichment. A plugin system allows different ways for Tatiana replayables to be visualised. We provide two sample *visualisation plugins* which we detail below. Such plugins are *Eclipse editors*, which, in the Eclipse rich client platform, allow the modification of a specific file type. Visualisation plugins must meet the following requirements:

- Know how to display any replayable (or use the help of the user and introspection to determine whether a given replayable type can be displayed) which is read using the appropriate API (cf. Appendix II).
- Implement the API which allows enrichments to be integrated (cf. Appendix III and §4.6)
- Be clients of the API which allows synchronisation and replay to exist (cf. Appendix IV and §4.7).

Visualisation plugins are granted complete freedom in access to the workspace and in what they do with the data. As such, they *could* also be used to generate complex transformations

[17] The most expressive computational devices are "turing-complete". Any computation which can be performed using a turing-complete language or device can be performed in any other. Conversely, if a result cannot be computed with such a language, it does not belong to the class of results which can be computed. In other words, replacing XQuery with another language would not grant greater expressive power – although it might reduce verbosity in certain situations.

and aggregations, such as calculating state-transition graphs or other forms of process analysis and data mining such as those presented in §1.4.2.

From a user's point of view when a replayable is selected, it can be opened by double click (presenting the default tabular view) or, can be "opened with…" the list of visualisation plugins which exist. Each replayable visualisation is contained in its own tabbed frame (using Eclipse's default functionality of opening each document editor in a new tab). Such frames can be accessible as a normal tabbed view or can be partially tiled, allowing several visualisations to be examined side-by-side (cf. Figure 4-3 and Appendix VI, which provides a user manual and shows more screenshots of Tatiana).

4.5.1 Tabular visualisation

The default visualisation is the tabular visualisation. This is used for editable replayables and for presenting the results of transformation. This visualisation presents data in a spreadsheet-like table where each event is presented as a row and each facet is presented in a column whose title is the facet name. A layout mechanism allows users to decide which facets should be displayed or not (e.g. by default the src-anchor facet is not displayed) and how each data type should be displayed (formatting, colour, etc.). For instance, by default only the hours/minutes/seconds of the start of the interval in the time facet are displayed. Multiple facets can also appear in the same column, in which case they are concatenated if they are not mutually exclusive in different events. The same facet can be visualised in different formats in the different columns (e.g. one column can show the begin time, another the end time and a third the date, all extracted from the time facet).

Facets in editable replayables and facets provided by enrichment are represented by editable cells in the table. The API for replayables transparently applies modifications to enrichment facets to the enrichments themselves (i.e. what appears to be the modification of one of the columns of a replayable is in fact a modification of the enrichment which provided that column).

4.5.2 Scoresheet visualisation

Scoresheet visualisation is a configurable graphical timeline, based on the idea of a musical score where each note is placed and shaped to inform about how it should be played. Each event in the replayable is represented by a graphical element (cf. Figure 4-3). The properties

of this graphical element are linked to the facet values of the event by *rules*. These rules can be defined incrementally to be adapted to a given need. Rules can be saved and applied to the scoresheet visualisation of other replayables. A rule takes the form of a (facet-name, facet-value, property-name, property-value) quadruple. The semantic of a rule is as follows: for each item, if the facet whose name is "facet-name" has the value "facet-value", the graphical property "property-name" of the corresponding graphical element is assigned the value "property-value". Facet names and facet values can also be replaced by wildcards.

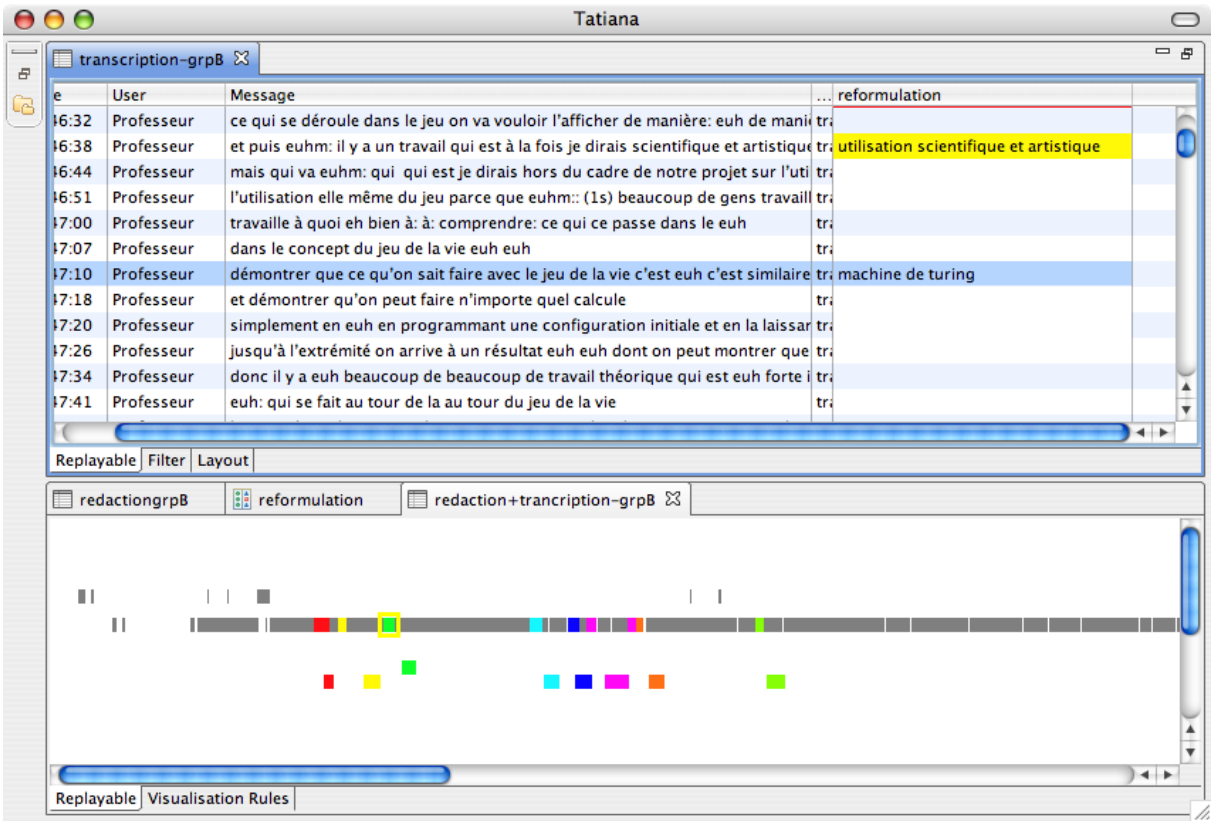


Figure 4-3 A replayable visualised in two different ways in Tatiana: tabular visualisation (top) and scoresheet visualisation (bottom)

One rule which is present by default and immutable is the association of the horizontal position of a graphical element with the time of the represented item and its width with the duration of the interval. The other graphical properties which can be assigned are visibility, vertical position, relative vertical position, size (height for rectangular shapes, diameter for others), shape and colour. A rule assigns numerical expressions to the value of those properties. These numerical expressions can be literal or use special variables which designate the values of the various facets of the item. A non numerical value uses the index of the value in the list of values taken by that facet. Colours must be expressed in hexadecimal notation (*red, green* and *blue* are special variables associated with 0xff0000, 0x00ff00 and 0x0000ff

respectively, allowing any colour to be expressed as a linear combination of these colours). A single rule can thus be used to place the representative of each event on a different line or colour it according to its user facet. A rule assigning the relative vertical position adds the value of the expression to the current vertical position of the element. Two rules can thus be used to place the six combinations of two three-value facets on six different lines.

A zoom functionality which only applies to the horizontal time axis is implemented on scoresheet visualisations (i.e. vertical spacing remains constant, regardless of zoom).

4.6 Enrichment

In Tatiana, each enrichment type is enabled by a plugin specific to that type. Like for visualisations, these enrichment plugins are Eclipse editors but, unlike visualisations, they only enable the edition and viewing of their own kind of analysis. Opening an enrichment file will expand the relationships contained into each open replayable and closing it will factor them out again. Relationships contained in an enrichment describe the events they refer to by using the src-anchor facet. We provided three sample *enrichment plugins*, detailed below and a basic implementation which can be extended and completed to create new kinds of enrichment. Enrichment plugins must meet the following requirements:

- Visualise, in some way, the relationships contained in the enrichment files they are designed to open.
- Make these relationships available to the Tatiana environment via the enrichment API (cf. Appendix III).
- Provide a graphical widget for editing the relationship attached to a given event. This graphical widget is usually a dropdown dialog or a text field. Visualisation plugins choose whether or not to make this widget visible to the user (i.e. whether the user can edit the enrichment from that particular visualisation). The tabular visualisation enables this edition, thus ensuring that there is at least one visualisation from which enrichments are editable.
- Be clients of the API which allows synchronisation and replay to exist (cf. §4.7),

From a user's point of view, the type of analysis (the plugin used) must be chosen when a new enrichment file is created. When an enrichment is created, or opened, a new tab is opened in Eclipse, from which it is possible to save changes to that enrichment. Within this tab, there

may be an interface for visualisation and/or editing the relationships recorded in the enrichment (or of some other aspect related to the analysis). In the tabular view, each enrichment is presented as a single column[18], with each cell containing the widget designed to create and edit that particular relationship for the event represented in the corresponding row. The title of this column is the name of the analysis. Correct namespacing of analysis names and facet names is currently not checked (i.e. there is no mechanism in place to verify that an analysis called X does not clash with a facet already called X).

Because of the file-based nature of Tatiana, enrichments are currently only "expanded in" for visualisation (modifications are instantly propagated through all open visualisations). For transformation, a mechanism which both knows how to expand enrichments in and to factor them out afterwards has not yet been implemented (i.e. for a script to know how to use the "virtual" facets provided by enrichment, they would have to a) be present and b) not be stored explicitly in the result of the transformation). This can be circumvented by saving a replayable as an editable replayable (implemented to optionally expand open enrichment columns in as permanent facets) or by first running the replayable through a script which expands in a given enrichment. In both cases however, the resulting enrichment facets are not factored out from the result after transformation (another script can be applied to perform this operation).

4.6.1 Categorisation and annotation analysis

The enrichment plugins for categorisation and annotation behave in a mostly identical fashion. They both record, for a given event, a single value associated with that event. For annotations, that value is a string that is entered into the textarea widget provided by the annotation analysis plugin. In the tab corresponding to a given annotation analysis, the list of annotations is shown.

For categorisation, the assigned value is a string, chosen by a dropdown widget provided by the categorisation analysis plugin (cf. Figure 4-4). The fields of this dropdown widget are populated through the analysis tab, where the list of categories can be edited. Each category can also have an associated colour, which can be used by visualisations. Lists of categories can be saved and reused in other analyses.

[18] In the current state of implementation, each relation may only provide a single additional facet and refer to up to two observed events. This is largely because of the difficulty, from an HCI perspective, of creating a user interface which would allow the full replayable model to be implemented.

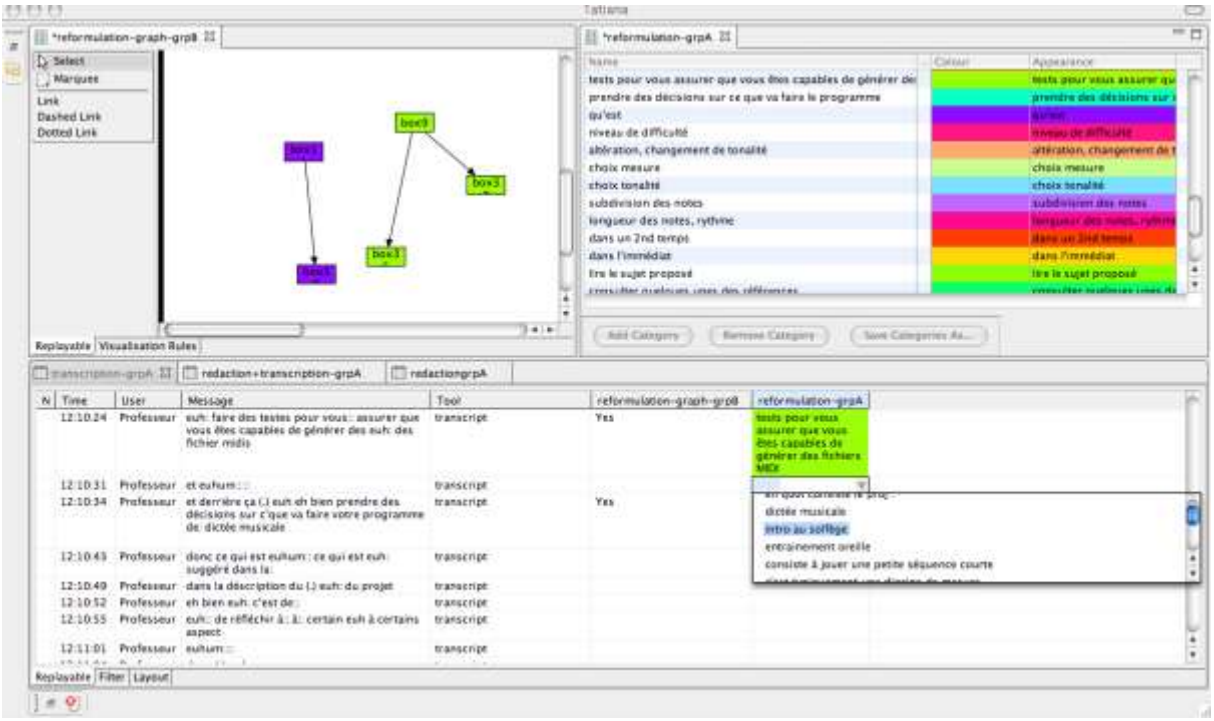


Figure 4-4 The interface for categorisation. Top right: the analysis tab with an editable list of categories for that analysis. Bottom: Tabular visualisation of a replayable with selection of a category for an event. Top left: a graph analysis which uses the colours provided by the categorisation analysis.

4.6.2 Graph analysis

The graph analysis plugin allows links between events to be created. For ease of implementation (there is no obvious user-interface mechanism for creating a link in a tabular visualisation), the editing widget provided by such an analysis is a dropdown specifying whether a given event should be included in the analysis or not. The tab for the analysis visualises all the events which are included in the analysis as boxes. Links can be created between these boxes. The links can be labelled and can be dashed, dotted and solid, however no further mechanism for the classification of links currently exists. These links are then available to all visualisations, which can choose (or not) to display them. The scoresheet visualisation displays such links, but does not allow their edition (cf. Figure 4-5).

For ease of navigation in the created graph, graphical analyses are synchronised with the rest of Tatiana.

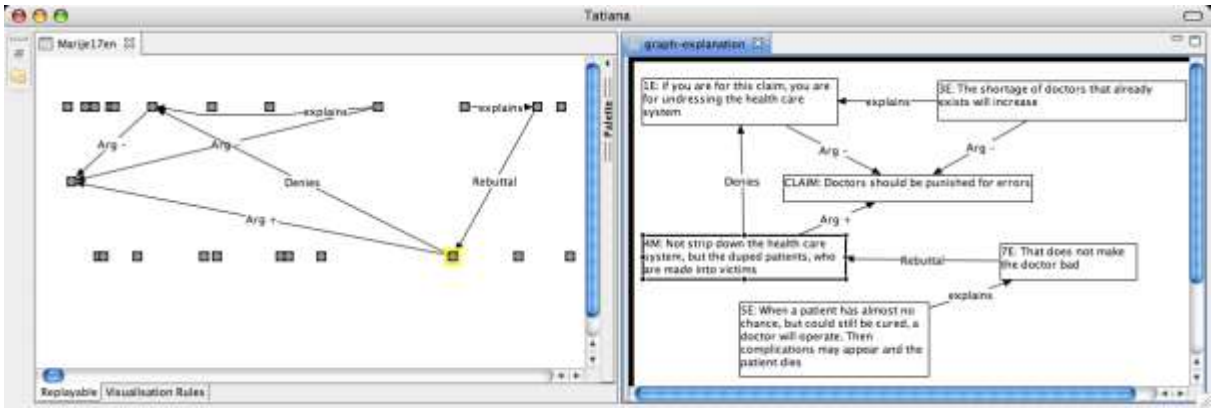


Figure 4-5 Graph analysis showing the reply structure of a chat as described by an analyst (right). This structure can be seen imported on a scoresheet visualisation (left). (data from a study presented in Amelsvoort, Andriessen, & Kanselaar, 2008)

4.7 Synchronisation

The last feature of Tatiana is a ubiquitous synchronisation and replay mechanism. In Tatiana, all open visualisations and enrichments are synchronised. This means that when the graphical element(s) for an event (or several events) is selected in one component, information about this selection is sent to all other components. The information which is sent includes two parts, at least one of which must be present: a time interval and a set of selected src-anchors. This is due to an unresolved dilemma between simultaneity and identity – sometimes synchronisation is used to go to the correct position in time; at other times, it can be used to identify how the same event is represented in different replayables. Clients of the synchronisation API must meet three requirements:

- Send synchronisation notifications when a graphical element matching an event or set of events is selected.
- Highlight the corresponding events when a synchronisation notification is received.
- List the timestamps for which the client wants to be notified during replay (typically all those of all the events present in the visualised replayable).

A remote control uses this synchronisation mechanism to allow replay. The remote control exploits the list of timestamps provided by the various components to be synchronised for two purposes: the first is to create a timeline on which intervals can be selected; the second is so that, during replay, appropriate synchronisation notifications can be sent (e.g. if the remote control knows that events exist at 4, 7 and 8 seconds, when play is pressed, it internally

"counts" the time and sends out notifications at those times to the components which requested notification). The remote control can also be used to set the speed of replay.

The tabular and scoresheet visualisation plugins completely implement synchronisation, sending notification whenever an event or set of events is selected and highlighting the corresponding events when a synchronisation event is received (cf. Figure 4-6). In editable replayables, this mechanism is also used so that intervals selected in the remote control or in other replayables can be used to create new events.

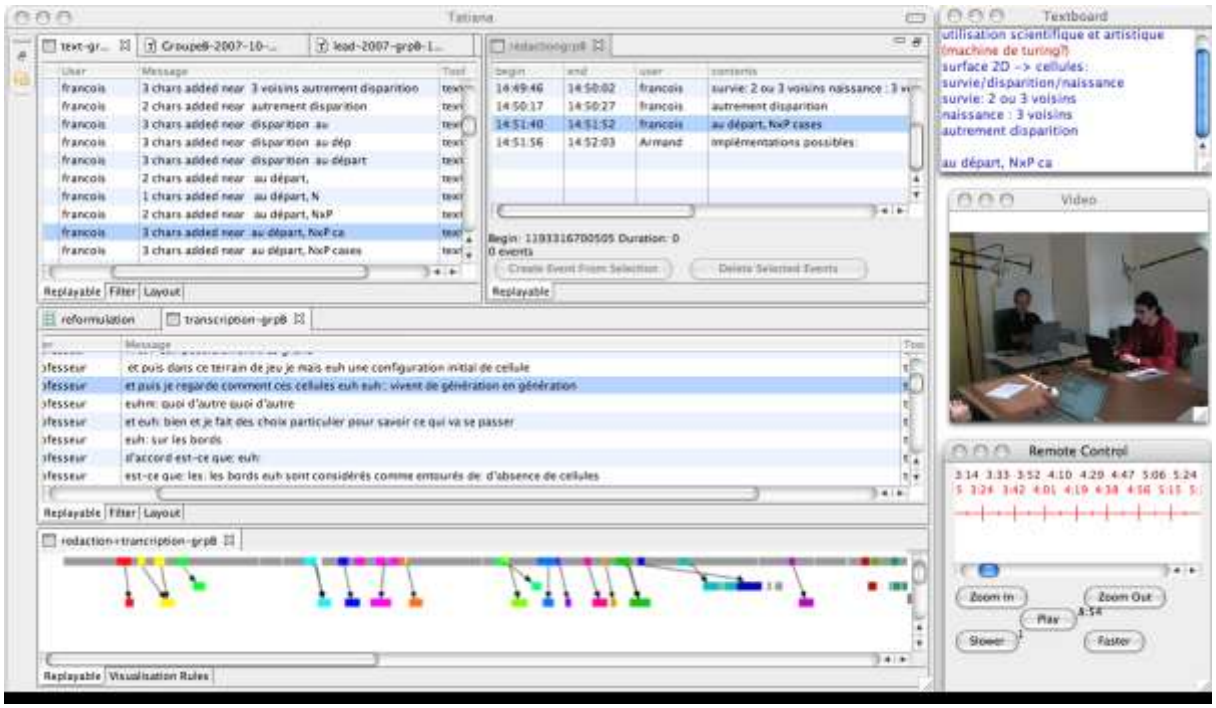


Figure 4-6 Bringing it all together. Synchronisation of several Tatiana visualisations, including three tabular visualisations (top, left and centre), the DREW external replayer (top right), a video player (centre right), a scoresheet visualisation (bottom left), and the remote control (bottom right).

The graph analysis plugin does not know about timestamps, only src-anchors. It therefore ignores synchronisation notifications which only provide a time interval. It also can only send notifications with src-anchor information and therefore does not interact with the remote control. The other two analysis plugins completely ignore synchronisation notifications and send no events of their own.

External replayers implement synchronisation by timestamp only (as they don't necessarily know about Tatiana's anchor mechanism). They also implement a replay API, choosing whether to manage their own time during replay (receiving just play, pause and speed information) or whether to use the remote control's "replay by notification" mechanism. In the former case, they should at least request notification when their first date is met. For

example, a video might only start after 25 seconds of digital trace. If play is pressed at the beginning, the play command will then only be sent to the video after 25s.

External replayers communicate with Tatiana via a protocol which uses XML-RPC. External replayers must set up an HTTP server on a port specified by Tatiana. This server then responds to synchronisation notifications (potentially including the supplementary play, pause and set speed) which are sent to it. External replayers cannot currently send synchronisation notifications back to Tatiana. We provide a sample implementation of such a server in Java. This implementation was adapted to create a replayer for CoFFEE (De Chiara *et al*., 2007), the software used for computer-mediated face-to-face discussion in the LEAD project.

4.8 Conclusion on implementing an environment for manipulating replayables

In this chapter, we presented the general architecture of an environment for manipulating replayables called Tatiana. We showed how the model for replayables presented in chapter 3 is implemented. We described the data model used in Tatiana and explained how the key operations defined on replayables interact. Automatic transformations are implemented through filters which make use of scripts. New scripts and filters can be created (respectively by users with and without programming experience). Manual segmentation and grouping can also be performed. Traces can be visualised in external replayers and replayables in the Tatiana data model can be visualised using any visualisation plugin. Two visualisation plugins are currently available, providing a tabular and a graphical timeline view. An API is defined for creating new kinds of visualisation. Replayables can be enriched through different kinds of analyses such as the already existing annotations, categorisations and graphs. An API is defined for creating new kinds of enrichments. Finally, visualisations, external replayers and enrichments can be synchronised and replayed together.

5 Results and perspectives

5.1 Introduction

In this chapter, we evaluate and discuss the work presented in the previous three chapters. We begin by presenting a number of case studies in which Tatiana has been used by various researchers, along with other ways in which Tatiana has been related to existing work. These case studies have been used as feedback during the iterative development of Tatiana, of our understanding of analysis through replayables, and of the replayable model. We then evaluate our work with regard to the criteria laid out in §1.6 and summarised in Figure 1-24. Finally, we discuss various research perspectives opened up by our work in the field of trace engineering.

5.2 Case studies

By its nature, the ecological use of an analysis tool to answer real-life analysis problems can only yield radically different uses of this tool. These differences are such that applying a formal analysis grid to each usage case or attempting to design an experimental protocol which would be followed during each usage case is not realistic. As such, we consider the uses of Tatiana which we have observed as case studies, with each case study iteratively contributing to our understanding of analysis, to our definition of the notion of a replayable and to the model we have designed and implemented for representing replayables.

Our observation of analysis in practice has been made difficult by several factors.

- Nearly every analysis problem (loosely defined as a data corpus and a set of research questions) is unique.
- Analysis is not something which can be simulated and, for a variety of reasons, is not always something which can be performed on command (in other words, agreeing on a date to come and watch an analyst perform their analysis is extremely difficult).
- Analysts need some time to adapt to a new analysis tool

While the description of and adherence to grounded methodologies is one of the goals of research, allowing the replication, understanding and validations of results, each analysis problem presents some unique characteristics. As such, it is challenging to determine which elements found in a given case study are unique and which are revealing of analysis invariants. In our work, we assume that all the analysis moves (or desired analysis moves) which we have observed are grounded in an analytic tradition and, as such, are likely to be done again at some point (and cannot, therefore, be ignored).

In order to perform an analysis, this analysis must exist within a research framework which is familiar to the analyst. It would be very difficult to come up with an analysis problem which could be used in an experimental situation where different analysts would be asked to solve the same analysis problem without also begging the question by framing the problem in such a way that a specific methodology should be applied. Furthermore, the analysis process tends to be something which is the fruit of a long period reflection and potentially many "false starts". Combined with the various other institutional requirements faced by researchers, setting a date to observe analysis *in situ* is almost not realistic.

All users of software have a certain mental model of how the software behaves. Existing analysis tools offer certain limitations on the kind of analyses which can be performed with them. Many of the analysts we have observed have been more familiar with analysis in Excel, or performed by hand, and as such do not yet exploit the range of possibilities offered by Tatiana.

In our case studies, we observe the instrumentation process (Rabardel, 1995), whereby analysts apply the tool we have built to the activity of performing their analysis. In doing so, this exhibits the gap between this activity and our understanding of it, as embodied by the tool we provide.

These case studies come from three main sources. The first is our own analysis work in collaboration with the ICAR laboratory in studies related to the LEAD project. The second is the LEAD project itself and our partners' uses of Tatiana. Finally, various new analysts wishing to use Tatiana emerged through other collaborations and demonstrations of our work.

For each case study, we present the means we used to gather information, the aspects of the research questions and situation which are necessary to understanding what analysts wished to do with Tatiana, the analytic artefacts which they created or wanted to create and how we used this feedback to improve our work.

5.2.1 Face-to-face collaborative note-taking

A lot of the development of Tatiana was motivated by our own analytic needs in analysing our own corpora in collaboration with the ICAR laboratory and in connection with the LEAD project. The corpus on which we have worked most extensively concerns collaborative note-taking. The information presented in this section stems from our own experience and covers usage of Tatiana which began in May 2007 and is still ongoing.

Situation. We observed nine dyads over a series of three to five meetings with their tutor, in the context of a programming project in an introductory programming course. These meetings happened in a face-to-face computer-mediated setting where participants could communicate verbally and had access to a chat and to a shared text editor – using the DREW software (Corbel *et al.*, 2003). The students were encouraged to take notes in the shared text editor. The traces collected during these observations were video to record dialogue and gesture, and the digital traces of the chat and the shared text editor. We were interested in observing and describing the multimodal reformulation which occurred, particularly that from oral discourse to written notes. In particular we looked at the linguistic nature of reformulation[19] and its impact for pedagogy and in determining how we can improve the benefit of these tutoring sessions for the students.

Analytic artefacts. Video is a replayable which presents events in continuous sampling and whose facets (the image and sound component) necessitate the existence of a machine to translate their representation. The trace of the chat is a replayable whose events have discrete sampling, produced each time a user sends a message. The facets of a chat event are the user name (identifying representation) and the text that was sent (textual representation). The trace of the shared text editor is a replayable whose events have continuous sampling (every second, provided a change has occurred) and whose facets are the user name and the complete content (at that point in time) of the shared text editor. This facet informs us on the state of the text editor but does not directly tell us the difference between the current state and the previous state, making it hard to infer the action that happened. Such a trace is easiest to understand with a machine: the DREW replayer (cf. §1.4.4.3).

In order to explore reformulation, we needed to create a replayable (or set of replayables) which represents units of expression (in textual, chat or spoken medium) in a granularity such

[19] For example attempting to find reformulations patterns which are similar to the transformations described in work on textual genesis (e.g. Grésillon & Lebrave, 1983, for structural transformations; Lebrave, 1989, for linguistic transformations).

that reformulation links made sense, allowing us to make these links explicit and examine them in greater detail. For the chat (little-used), we left the message-level granularity untouched. For the dialogue, we set it at utterance-level via a manual transcription (with occasional breaks during pauses and very long utterances). For the shared text editor, however, we had to establish the notion of a *writing unit* which, informally, consists in a phrase or sentence in the written notes which forms a coherent semantic unit. In order to transform the trace of the shared text editor into a replayable containing writing units, we performed several intermediary steps. First, we applied an automatic transformation which adds a facet to each event, describing the difference between the current state and the previous state in a form which is understandable by humans (this was done as part of the import into Tatiana). We then created a replayable through an automatic segmentation of the trace based on inactivity periods (this segmentation had only the user facet). With the help of these replayables, we created a manual segmentation into writing units which included a user facet and a facet containing the text which constituted the writing unit. In order to have a visualisation on which both utterances and writing units can be seen, the transcription and writing units were merged into a single replayable. The set of created replayables can be seen in Figure 5-1.

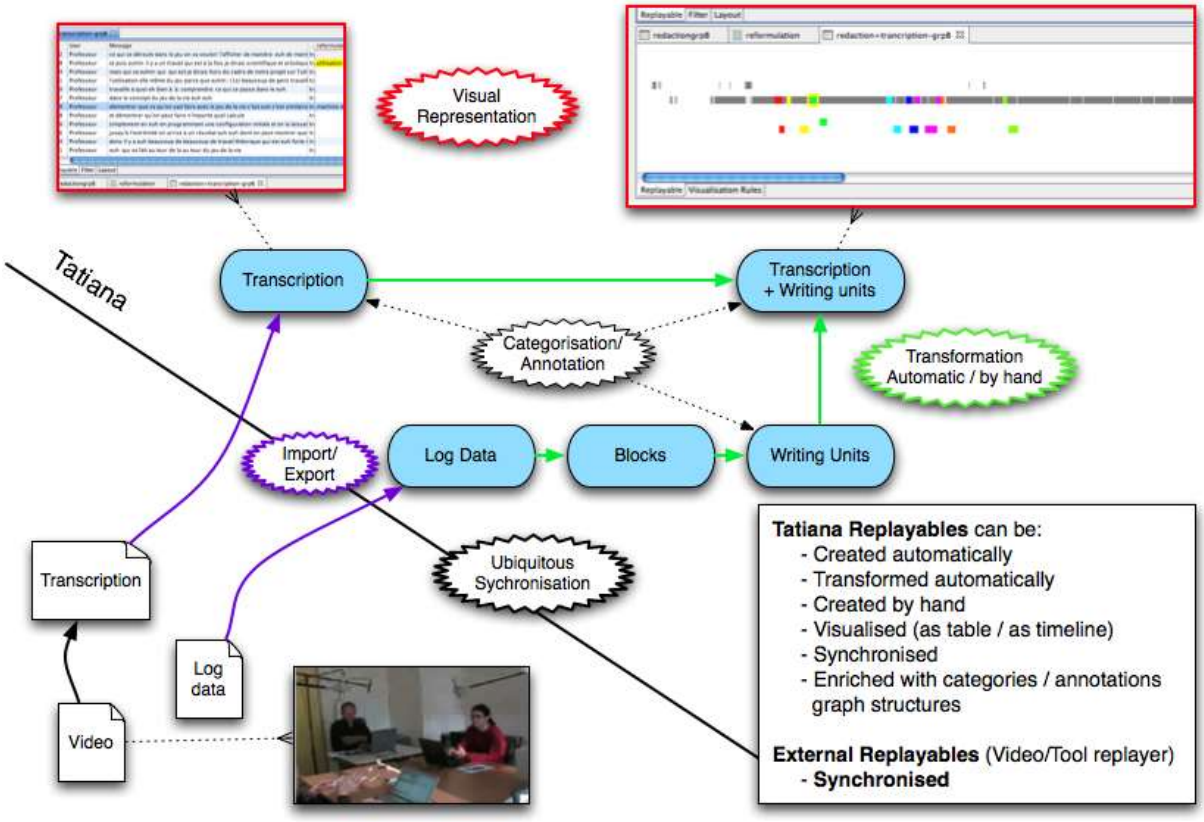


Figure 5-1 Process used to analyse reformulation

To show reformulation, we first created a categorisation analysis with one category per reformulation and a colour associated with each category. We enriched both the reformulated utterance and the reformulating writing unit with this category, using tabular visualisations of the writing units and the transcription, synchronised with the video and the digital trace viewed in the replayer. We then created a graph analysis, drawing each reformulation link. The combined replayable (writing units and transcription) was then visualised as a scoresheet with reformulation shown both through the links of the graph analysis (non present in Figure 5-1, but shown in Figure 5-2) and the colours of the categorisation analysis. Events from the transcription were placed on the top row and writing units are placed in the bottom row, each row being subdivided into individual rows for each user. In order to distinguish reformulation by different users, we also adapted the rules in order to create a different visualisation with one user's writing units on the top row, the utterances in the middle and the other user's writing units at the bottom. Outside of Tatiana, we completed this replayable with labels and other information (as shown in Figure 5-2). We have used these various replayables in two preliminary descriptive papers (Dyke *et al.*, 2007; Lund, Dyke, & Girardot; 2009).

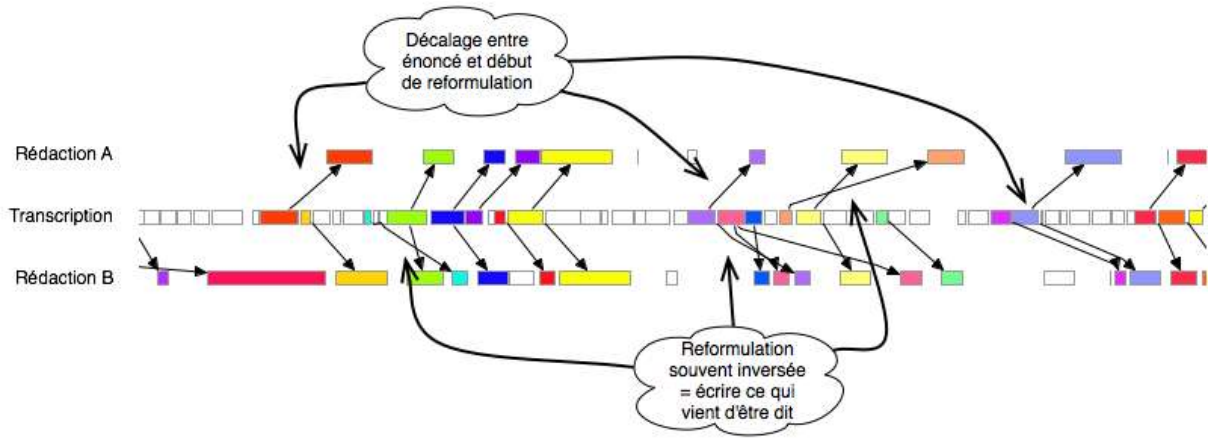


Figure 5-2 Tatiana replayable exported to an image file and completed with labels and comments.

Relevant feedback. As this analysis (and others like it previously done at ICAR) was one of the main sources of inspiration and even motivation to implement Tatiana, many concepts were drawn from this to improve our understanding of analysis (for instance that we wanted to create different replayables for different purposes and not one single, continuously improving replayable) and of replayables (for instance that we wanted graph structures to be able to span across replayables rather than having to be created from within a single replayable). The choice of editable replayables, the scoresheet visualisation and the graph analysis plugins stemmed directly from this work, as did the link with the DREW replayer.

While scoresheet visualisations allow us to create many of the visualisations we want, they are still missing labels and textual annotations. Reusable transformations were inspired by noticing that, if properly thought out, our script for detecting pauses could be made generic and applicable to any replayable. Finally, this case study allowed us to verify that analyses and replayables can indeed be shared with other users of Tatiana and exported to other data formats.

5.2.2 Paris case study

This case study was performed as part of the LEAD project, within which Tatiana was one of the deliverables: a tool to assist the analysis of face-to-face computer-mediated learning situations. Our observations in this case study consisted of informal discussions throughout the LEAD project and a two day on-site visit during which we observed one of the classroom studies of the Paris team, discussed their analysis objectives and performed video recording and screen capture of a novice's use of Tatiana. The usage of Tatiana in this case-study was limited to the months of June and July, 2008 although the LEAD project (and our interaction with researchers within it) extended from January 2006 to December 2008.

Situation. In this study, the researchers were interested in observing the kinds of interactions which emerged during the use of CoFFEE (the discussion tool created as part of the LEAD project, cf. De Chiara *et al*., *op. cit.*) in the case of face-to-face collaboration (Bernard & Baker, 2009). The students they observed were working in groups of four, with each group made up of a pair of dyads, one computer per dyad. As such, face-to-face discussion was available both within a given dyad and between dyads (dyad pairs were placed in front of adjacent computers). Computer mediated discussion was available between dyads. The discussion tool made available to the students was a collaborative argumentation graph editor. The researchers wanted to use Tatiana to pinpoint the kinds of activities performed by each dyad in order to use the alignment of these activities across dyads as evidence of dyad-dyad collaboration.

Analytic artefacts. Because of the cost of transcription and the fact that activities span across many interactions, the aim was to directly use the remote control to select intervals on the timeline in order to create replayables containing segmented activities of each dyad using the replay of the video and the digital trace as a guide. These activities could then be categorised and a scoresheet visualisation of a replayable combining the activities of two dyads could be used to establish the nature of the collaboration between dyads. In practice, the CoFFEE

replayer did not exist at that point in time and it was not possible (at the time) to create events from a selected interval in the remote control in Tatiana. Because of the other analysis approaches used by the Paris team (comparative study of the graphs created as the students appropriated CoFFEE over a series of experiments, and of the contents of these graphs and discussion in other media), analysis with Tatiana has not yet been performed.

Relevant feedback. This case study was the first outside of our own research team and taught us a lot about the difficulties in studying analysis with Tatiana. We had previously asked the researchers from the Paris team (and other teams in LEAD) what they wanted from an analysis tool, but until sitting down with them and attempting to use Tatiana, several shortcomings had not been apparent. The most obvious of these was that the Paris team had not intended to work from a transcription (previously performed in some other tool and then imported into Tatiana) and as such, it was necessary to be able to segment events from a timeline within Tatiana. The dialogue between segmentation, category definition and category application confirmed both the necessity of being able to adapt categories over time and the impracticality of considering analysis as a process wherein steps are executed in a consecutive fashion. We also were able to pinpoint many usability problems in Tatiana and confirm the comparative difficulty in working with traces in the absence of a replayer.

5.2.3 Utrecht case study

The Utrecht case study was also part of the LEAD project. Our observation consisted of informal discussions, a two day on-site visit, during which we recorded a video of discussion of previous analysis strategies and of initial usage of Tatiana, and a report written three months later by one of the researchers, detailing his use of Tatiana over the period from July 2008 to September 2008.

Situation. In this study, researchers were interested in the discussion that can be performed by using a shared argumentative graph editor, compared with discussion using graphs created with paper and pencil. They hypothesized that discussion in CoFFEE leads to more equal participation. A method used in previous analyses consisted in the manual creation of a graphical timeline (cf. Figure 5-3) which shows different discussion “threads” in the argumentative graphs as evidenced by the positioning of nodes in trees and in space. Students’ navigation between these threads is then shown by lines sequentially tracing their activity. Another method consisted in placing only “box creation” events in Excel and re-

ordering them into threads (rather than chronologically) in order to create a "discussion" which can then be analysed using discourse analysis methods.

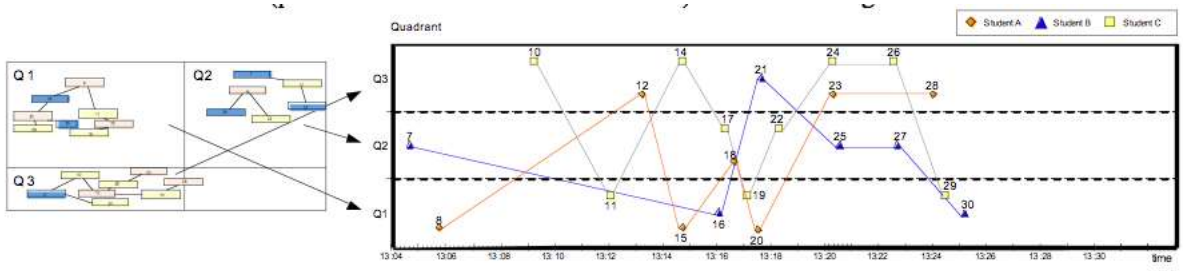


Figure 5-3 Discussion produced in three quadrants of the graphical argumentation space (left) is represented in a graphical timeline (right) with one row per quadrant and identical shapes combined with lines to show the movement of each student across the quadrants. (van Diggelen, Jansen, & Overdijk, 2008, p. 6)

Analytic artefacts. In order to compare face-to-face discussion combined with computer-mediated discussion with face-to-face discussion without computer mediated discussion, a replayable was created for each group from the transcription of the face-to-face dialogue. For groups with computer mediation, additional replayables were created from the digital trace, which were then edited by hand, deleting events which could not be interpreted as dialogue turns and re-ordering these events into "discussions". These replayables were then coded to show different forms of participation (task-related and social-emotional). This coding was then exported to Excel for statistical analysis.

Relevant feedback. In this study, we knew that the requirements we gathered during the visit would have to be quickly implemented – leading to the creation of editable replayables with events which can be ordered and deleted. However, at the time we were not able to provide links between events in the graphical timeline and even now, these links can only come from a graph analysis – we have not yet provided the means to automatically create analyses or to show links which do not come from enrichment. One way in which this could be done would be to create a new enrichment plugin which automatically creates relationships between subsequent actions with similar properties, defined by the user. In this case study, we were also able to confirm the relevance of allowing export to Excel for aggregation operations to be performed.

5.2.4 Nottingham case study

Our last case study within the LEAD project came from work with the Nottingham team. Their analyses were primarily statistical in nature. Because of this, our observation stems exclusively from informal discussions during meetings and by email, regarding their requirements. Tatiana was used in Nottingham mainly during the month of September 2008.

Situation. In order to examine the effect of anonymity in group discussion and voting in the classroom, a pedagogical situation was organised where various topics were under discussion using the CoFFEE software (for an exploratory study, see Gelmini Hornsby *et al.*, 2008). For each topic, a vote was first taken, followed by a discussion, followed by a new vote. This experiment was run in both anonymous and non-anonymous conditions. Researchers wanted to compare both the differences in voting and in discussion through statistical analysis of the two situations.

Analytic artefacts. The digital trace of the CoFFEE software contains all the events that happened during a session. In order to prepare for statistical analysis, votes which were not final needed to be removed and events had to be filtered by discussion topic. This was accomplished using transformations in Tatiana. Tatiana also facilitated calculation of certain simple aggregations such as number of words per turn, average number of words per turn and per student and contingency tables. All replayables were then exported to Excel for further analysis.

Relevant feedback. Based on previous experience of such data manipulation without support of dedicated software, Tatiana greatly facilitates the authoring of reusable scripts for performing certain common operations – because of our reliance on the Tatiana display format. This case study showed the importance of using Tatiana to handle replayables (something it is designed to do well) while allowing any replayable to be exported to another tool (designed to do another job well, such as statistical analysis). It also yields insight into how quantitative analysis can be used to feed qualitative analysis and vice versa (in this regard, Tatiana could automatically perform certain quantitative analyses, giving indications of which aspects of a corpus might be of special interest).

5.2.5 Collaborative design in engineering

In other projects in which the ICAR lab is involved, some colleagues have been studying how collaborative design in engineering is carried out. The information about this case study was obtained from participation in meetings regarding the analysis of two corpora over a period of several years and through usage of Tatiana between March and July 2009.

Situation. A common goal for these two corpora is to identify the argumentative mechanisms through which collaborative design in engineering unfolds. A result on the first corpus detailed a number of patterns of argumentation leading to the approval of one solution over another or to the improvement of a solution (Prudhomme, Pourroy, & Lund, 2008). The second corpus was the recording of a review meeting between two distant teams (one working on the design of a new truck and the other validating the possibility of constructing the truck on the factory floor) in which twelve design decisions were discussed. The researchers were interested in finding recurrent discussion patterns in the decision-making process (Cassier, in preparation).

Analytic artefacts. In the analysis of this second corpus, the video data was first transcribed and then coded according to two coding schemes, the first describing the subject of the utterance and the second describing the function of the dialogue act. This data was then imported into Tatiana. Contingency tables were used to attempt to find correlations between users and codes or between the codes themselves. A data mining algorithm to find the subsequences common to all twelve decisions was applied with no success (because this algorithm is commonly used to find common retail patterns across thousands of different buyers – in this case, there were too many re-occurring codes or "identical buyers" and too few decisions or "sequences of buying by a single user"). A script written for Tatiana to find the most common sequential occurrences was applied, but the results were found to be very difficult to interpret (how can we extrapolate that speaker A often talks after speaker B into some kind of generality?). Finally, describing the argumentative structure of the discussion with a graph analysis (cf. Figure 5-4) revealed some interesting patterns when this data was later visualised as a scoresheet (cf. Figure 5-5).

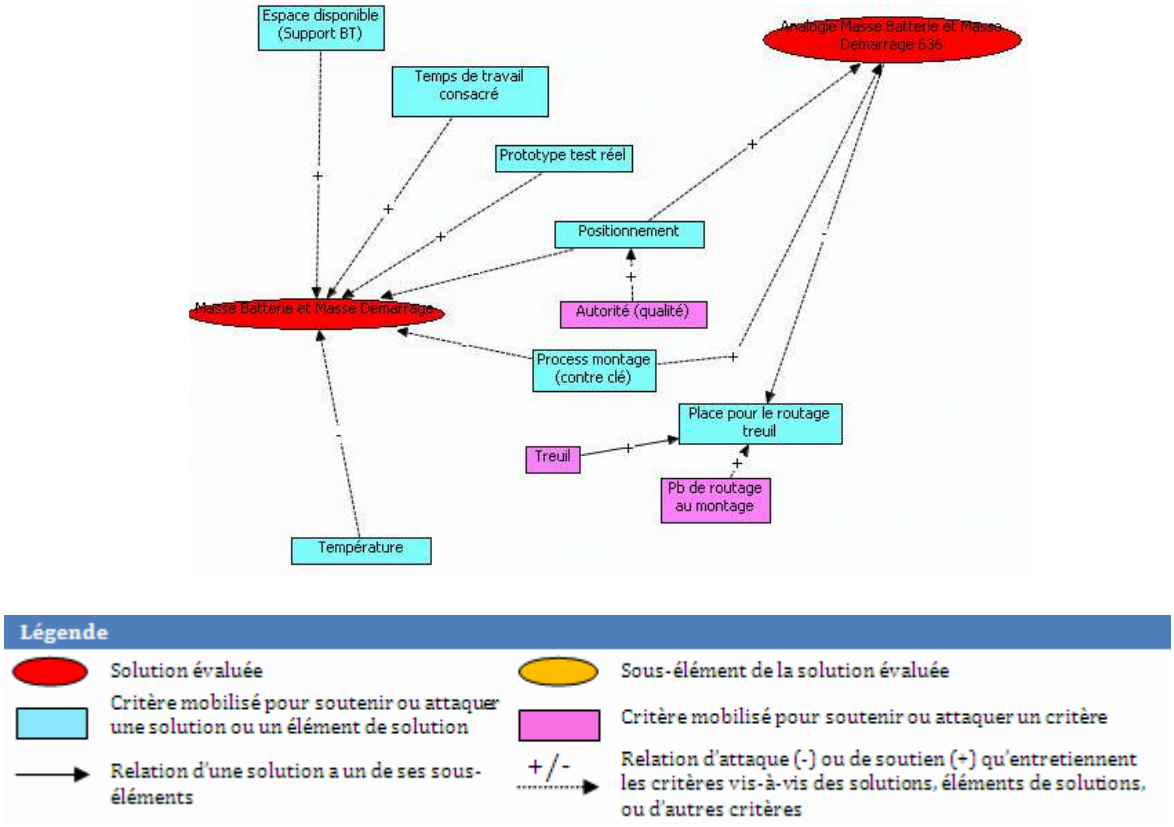


Figure 5-4 Two solutions are comparatively evaluated according to various criteria (Cassier, in preparation).

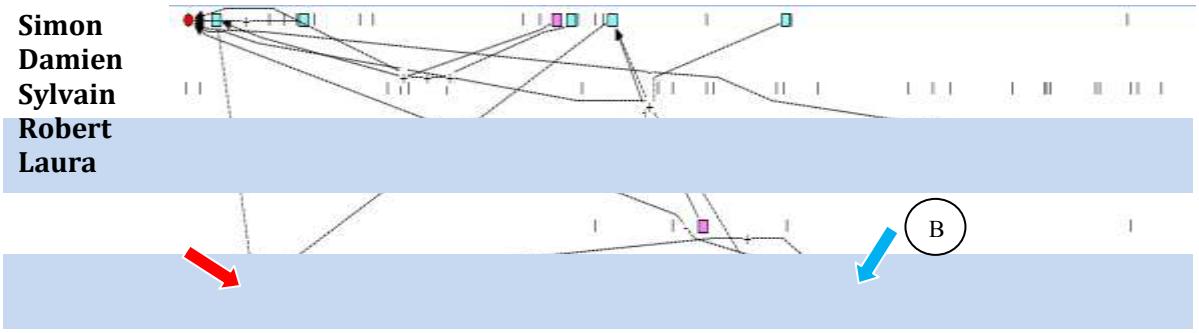


Figure 5-5 The graph presented in Figure 5-4 is now shown as a graphical timeline, with additional arrows and labelling performed outside of Tatiana, showing one of the solutions and the criterion which was pivotal in choosing that solution (ibid).

Relevant feedback. This case study highlighted the usefulness of being able to share analyses as, on the first corpus, three analysts from four different fields of expertise (mechanical engineering, collaborative design, socio-cognitive interaction and argumentation) contributed to an analysis which could not have been done by one analyst alone. While we believe that data-mining algorithms can produce some interesting results, it remains difficult to see how such information can be interpreted, particularly when each element of data is coded individually, rather than in relation to other elements. The results produced by making the

structure of the discussion explicit were encouraging and confirmed the value of graph analyses. Perhaps data-mining techniques could be applied to such structural analyses in order to find common patterns in these structures.

5.2.6 Describing the structure of chat dialogue

In a previous European project called SCALE[20], chat-only dialogue was compared with dialogue using both chat and an argumentative graph editor. We have begun to perform new analyses on this corpus by representing the implicit structure of the chat as a graph in order to be able to compare it with the structures present in the argumentative graph editor (cf. Figure 4-5, Chapter 1). This usage of Tatiana occurred during the months of October and November 2008.

Relevant feedback. This case is interesting in that it confirms once more the usefulness of the graph analysis and because, in spite of this corpus being over five years old, we were able to "resurrect" it with little trouble in Tatiana. We also explored the use of contingency tables as means to provide suggestions for more detailed qualitative analyses.

5.2.7 Tension and relaxation

This case study stems from work with colleagues with whom we collaborated to help them create the visualisations they wanted in Tatiana between October 2008 and January 2009.

Situation. The analysis in this case stems from the theoretical idea that, in order for successful collaboration to occur, there must be a balance of tension building moments (which can illustrate disagreement and serve to pinpoint issues which must be addressed) with tension releasing moments (which ensure that the people collaborating are getting along and that their emotions are not getting in the way of a good collaboration) ; stereotypically, too much tension will cause so much friction between participants that they will no longer be willing to cooperate and too much relaxation might be indicative of avoiding the important issues (Baker, Andriessen, & Lund, 2009). Our colleagues wanted a way of visualising a collaboration session so as to illustrate intuitively the nature of a session. They also wanted to be able to compare this with the depth of discussion about various topics.

[20] Internet-based intelligent tool Support Collaborative Argumentation-based LEarning in secondary schools, IST-1999-10664, funded under the IST programme, framework-VI.

Analytic artefacts. The data for this study was exclusively oral dialogue captured from video and transcribed. This transcription was imported into Tatiana and then coded in three categorisation analyses: the first distinguished tension raising utterances from tension relaxing utterances, the second described the concept addressed by each utterance, and the third qualitatively rated the "depth" of the contribution from one to three (depending on whether it contained concepts expressed in relation with each other and whether the utterance was argumentative in nature). Two scoresheet visualisations were then created for each session, the first showing topics by colour and depth of discussion by height of the event, the second showing tension and relaxation by colour and participant by vertical position of the event (cf. Figure 5-6). These visualisations were useful to characterise different kinds of discussion (e.g. a moderator who always raises tension without leaving time for emotions to unwind, or a discussion that goes well until a key moment when too much tension is raised) and to pinpoint relationships between tension-relaxation and depth of discussion which are worth exploring in greater detail.

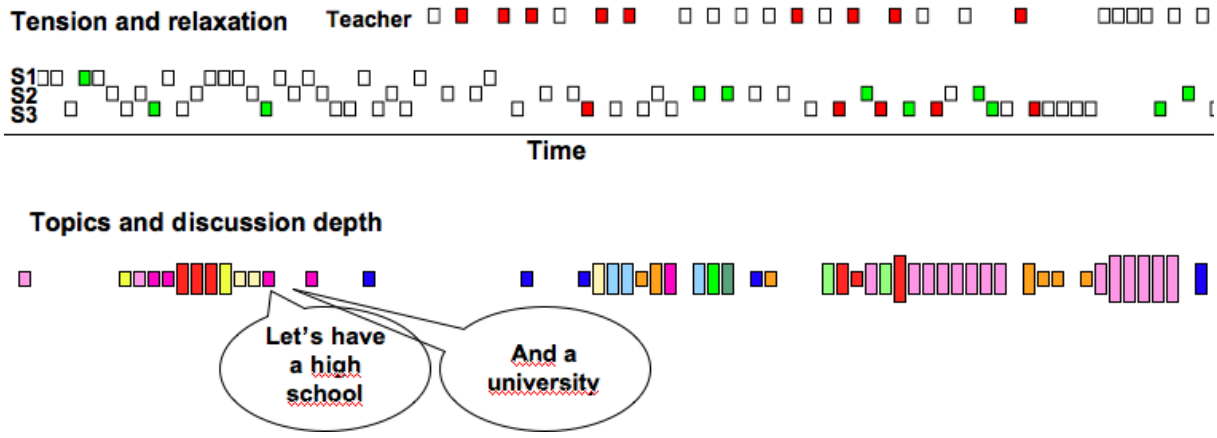


Figure 5-6 Two visualisations representing the same discussion. The first illustrates tension raising (red) and tension relaxing (green) interventions by the teacher and the three students. The second identifies topics by colour and the height of each element shows the depth of discussion (Baker, Andriessen & Lund, 2009).

Relevant feedback. This case study was the first large use of Tatiana to create visualisations outside of our own work (in the sense that the initiator of this use was not under any institutional obligation to use Tatiana). It is also particular in that while the corpus was collected in a learning situation, it was not at all computer mediated and the analysis focuses mostly on collaboration. It illustrates the variety of visualisations which can stem from appropriately chosen rules in the scoresheet visualisation plugin. It also served to pinpoint a number of usability problems with Tatiana. Some of these have been solved (e.g. keyboard access for easier categorisation) and others, such as the complexity of using the interface for

defining rules, have not. Finally, this case study shows how visualisations can be used to pinpoint interesting moments which might be worthy of a more detailed analysis.

5.2.8 MULCE

The researchers from the MULCE team cf §1.4.1.3 approached us to use Tatiana for two purposes. The first was a means for sharing corpora of online interactions (recordings of screen capture and audio complete with transcriptions of both) in a way that would be easy to read by other researchers. Second, they wanted to perform and publish some of their analyses on these corpora. On the MULCE platform, this data is published with a link to download Tatiana and an explanation for how to view the data within Tatiana. This work was performed between February and May 2009.

Situation. The situation under study uses an online conferencing environment which has oral discussion, chat, shared text editor and whiteboard facilities. The pedagogical context is a tutoring session between three students (French native speakers) and a tutor (English native speaker) during an English course. They are answering a quiz about interactivity, focussing on the analysis of an educational website. In the analysis, the authors address the notion of context and the extent to which different contexts are shared by the participants – most markedly in their choices of communication modes and media.[21]

Analytic artefacts. Because the conferencing environment used does not produce traces, the original recordings were screen capture and audio. Both of these media were transcribed by hand. The extract was broken up into three phases and the media coordinations were categorised according to the context to which they belonged (out of context, shared context with at least one other participant, negotiating context, etc.). This analysis was made explicit in Tatiana in order to make it publicly available. Our colleagues constructed two visualisations showing the two parts of their analysis (cf. Figure 5-7).

Relevant Feedback. This study provided useful feedback with regard to the usability of Tatiana. In particular, the authors of the corpus and analysis wrote their own documentation in supplement to that provided with Tatiana in order to ensure that their data would be accessible to anyone who downloaded it. More importantly, it is a significant use of Tatiana for sharing analyses with other researchers, confirming that certain analytic artefacts can indeed be represented within Tatiana. Furthermore, as shown by the constructed visualisations, this

21 This corpus is available online and is described at http://mulce.univ-fcomte.fr/metadata/doc/visu-corpus/copeas-T5_contexte.xml

enabled the creation of new artefacts from these analyses – illustrating the power of being able to capitalise on existing analyses. As a somewhat surprising element (after our prior experiences with scoresheet visualisations and their usability problems), the visualisations described above were created without prompting or assistance on our part (apart from the limited documentation).

Figure 5-7 Two visualisations of the same corpus. Both visualisations distinguish participants by colour. In the top visualisation, the three major rows correspond with the three phases. Within each of these, the minor rows indicate which communication media was used.

5.2.9 Blogs

In this case study, we were approached by a researcher from Denmark who wanted to use Tatiana to analyse data from blogs. Our collaboration mainly consisted of writing scripts to import this data into Tatiana and creating of certain specific aggregation calculating scripts. Usage of Tatiana was from November 2008 to April 2009 and will resume in 2010 as the researcher explores other kinds of online communities such as Twitter.

Situation. The data collected for this analysis was the posts and comments on two blogs with a high post rate (over 300 posts a year and between 20 and 100 comments per post) over a

period of six months. Our colleague was interested in examining the structure of the community around these blogs (Lomborg, 2009).

Analytic artefacts. Because of the large quantity of data, it was chosen not to order all the events in the replayables by date. The import script ordered posts by date and placed the comments related to each post in the rows below that post. Because of Tatiana's technical limitations, a replayable was created for each month (more than 2000 events in a replayable and Tatiana's visualisations become very slow – this is an easy problem to fix, but has not been a priority until recently). A first analytic move was the annotation of various posts and comments in order for the researcher to "appropriate" her corpus. Some replayables representing collections of comments of a similar nature have been created. Various aggregate statistics were also counted in order to see which users commented most, which posts were longest etc. Other scripts were written to determine the activity in terms of "time of day" (morning, afternoon, evening, night, etc). Finally, interview transcriptions were also imported into Tatiana and annotated.

Relevant feedback. Aside from some useful feedback about the usability of Tatiana and its technological limits, this case study serves to illustrate how Tatiana can be used for other kinds of analysis – in this case more of an ethnographic approach. While other data analysis tools exist for doing this work, Tatiana has the added benefit (which the researcher plans to use when she returns to the blog corpus) of being able to relate data back to a timeline, in order to show how a community evolves.

5.2.10 Knowledge forum

In this study, we collaborated extensively by email with researchers from Hong Kong who wanted to visualise data from Knowledge Forum (cf. §1.4.2.3) over the period of March to May 2009.

Situation. In this study, researchers were interested in relating the *knowledge building* epistemology for learning (cf. §1.2.1) with argumentation (Law, Lu, Leng, Yuen, & Lai, 2009). They coded data contained in a knowledge forum according to two categorisation schemes, the first related to knowledge building and the second to argumentation. They wanted to use this data to create CORDTRA-like visualisations (cf. §1.4.4.1) which would help identify patterns or correlation between the two categorisations.

Analytic artefacts. The data had already been imported into several Excel worksheets and coded there. The coding scheme used might more appropriately be called a tagging scheme whereby, for each message event, categories are not mutually exclusive, but cumulative. This posed problems for import into Tatiana, as the categorisation analysis plugin only provides mutually exclusive categories. The desired visualisation has each event represented by several graphical objects, one for each code. Again, Tatiana's scoresheet visualisation only provides the ability to represent each event by one graphical object, making this difficult. The workaround used was to create a replayable with several copies of each event, one for each category to be applied. This enabled both the import of the categorisations and the desired visualisation (as for each event, its duplicates provide all the needed graphical objects, cf. Figure 5-8). Interpretation of the produced visualisation is still underway.

Relevant feedback. This case study serves to illustrate the value in several choices made related to the concept of replayables and the design of Tatiana. First, the data coded in Excel is intrinsically related both to a single replayable and to a particular visualisation of that replayable, making its reuse for other purposes challenging. For this reason, we relate enrichments to the observed event for which the events in replayables (and more particularly in their visualisations) are only proxies. Furthermore, while Tatiana in its current state had to be "worked around" to produce the desired result, the underlying replayable concept is sound. What is missing is not core functionality but the correct visualisation and enrichment plugins. The only feature which is currently missing in Tatiana to implement these plugins is its ability to understand sets of facet values (existing in the replayable model but not yet implemented). Once that is added, visualisation and enrichment plugins can be created to meet the needs of this case study with no additional modification of Tatiana.

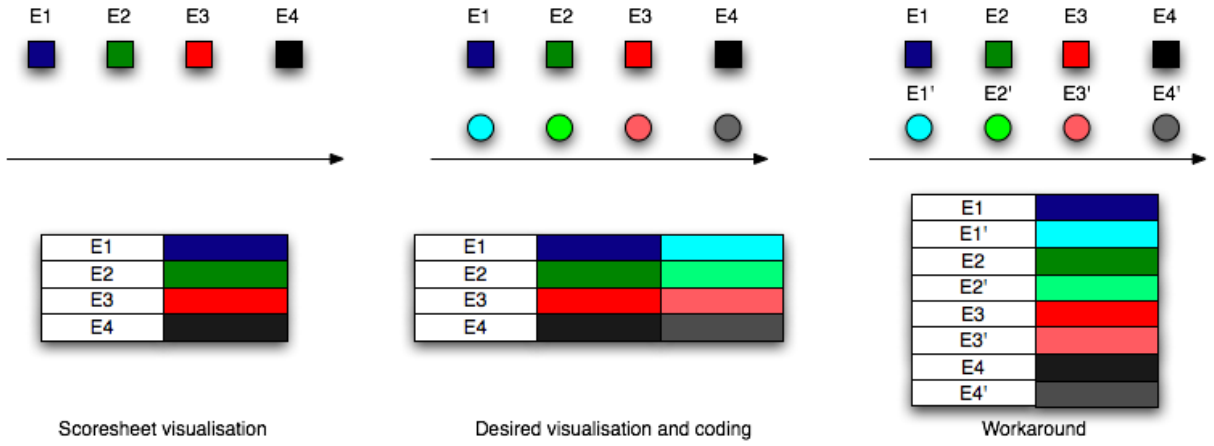


Figure 5-8 The missing visualisation plugin. Left, the relationship between data in a tabular view and the scoresheet visualisation (one graphical element per event). Middle,

the desired visualisation (one graphical element per coding feature). Right, the temporary solution (each event is duplicated and both codes are placed in the same column).

5.2.11 Tatiana interoperability

Aside from use in the MULCE project (detailed in §5.2.8), Tatiana is currently being used by various other researchers or teams of researchers. We note in particular two efforts at integration with other tools.

The first is the currently ongoing integration of KSV (Knowledge Space Visualiser; Teplovs 2008; cf. §1.4.2.3) as an enrichment plugin for Tatiana. The benefits of such integration are numerous. Tatiana can be used as an environment to synchronise a KSV visualisation with other replayables based on the same data (e.g. a scoresheet visualisation). KSV can use the enrichment API to use enrichments created in Tatiana as data for its own visualisations. KSV can expose the graph structure it is designed to visualise as an enrichment in Tatiana, available to other replayables. Finally, because of Tatiana's design for interoperability, any corpus which can be opened in Tatiana is a candidate for analysis in KSV.

The second is linked to Tatiana's ability to export to any format. The authors of Calico (cf. §1.4.4.2) have designed a tool for viewing and analysing any kind of forum. We have written an export script to Calico. Any forum-like corpus which can be opened in Tatiana can now be exported to Calico. We are currently exploring ways in which the analyses performed in Tatiana and Calico can complement each other and how analyses performed in one tool can be exploited or shared with the other.

5.2.12 Summary of case studies

In this section we have presented ten case studies of Tatiana use in eight labs and five countries. Tatiana was not necessarily successfully used in each case but, in its current state (modulo the implementation of new plugins), Tatiana can fulfil all the laid out requirements. Several other collaborations in the use of Tatiana are currently under way.

In all our collaborations, we are focussing our efforts in enabling as many useful analytic moves as possible to be performed in Tatiana (from simple moves such as transformations, to more complicated ones such as visualisations and enrichments), and enabling as many different kinds of data as possible to analysed within Tatiana. The end goal is to provide an environment for analysis within which it is possible to construct and document new analysis

methodologies which can be shared with others and upon which we can draw to develop our understanding of analysis.

5.3 Evaluation

In §1.6 we laid out the goals for this dissertation and the means by which we would evaluate whether these goals had been met. Having presented the different results of our work, we can now draw the necessary links between them to evaluate the individual and global result. Because of the iterative nature of our work, many of the criteria used in evaluation receive positive answers because we improved each of the three aspects of our work so that it would meet that criteria.

5.3.1 Replayables as generic analytic artefacts

In chapter 2, we defined the replayable as a generic analytic artefact used in performing certain kinds of analyses. We specified these analyses as those interested in some aspect of the process of human interactive activity. We defined the operations which can be performed on replayables as synchronisation and replay, visualisation, transformation, and enrichment.

5.3.1.1 Can replayables represent all the analytic artefacts and moves presented in the state of the art?

As the operations defined on replayables were extrapolated from the state of the art, the answer to the second part of the question is yes by definition[22], although it is possible that exceptions will be discovered in the future. With regard to the first part of the question, replayables represent a very specific kind of analytic artefact, which serves as a proxy for events which unfold over time. This specifically excludes artefacts such as statistical analyses and visualisations of aggregation data, particularly when compared with aggregation data obtained. As such, replayables represent a subclass of analytic artefacts which have interesting common properties and for which the existing models and software support were found lacking (a lack we hope to have remedied through our work).

[22] By this we mean that the iterative loop was run (over a period of several years) until we were satisfied that the operations we defined on replayables adequately represented the analytic moves in the state of the art.

Replayables enable all the analytic moves for exploratory sequential data analysis as described by Sanderson & Fischer (*op. cit.*), except comparison and calculation (which do not produce replayables). These are only accommodated insofar as they can be done with synchronisation and visualisation (for comparison) or transformation (for both), as explained in §2.5.5 and §2.5.6.

5.3.1.2 Can this definition of replayables allow and inform the design of a model for representing it?

The model for replayables presented in chapter 1 exists in direct relation with the operations defined for replayables (i.e. how the model enables these operations should be relatively straight forward). Another way to describe this would be to say that the defined operations can be implemented in an orthogonal way. This, we believe, is in marked contrast with the analytic moves as defined in other work, where, as we have shown, different analytic moves can be operationalised in one of several ways (e.g. comparison can at least be implemented via synchronisation, visualisation and transformation). Most of the decisions made in construction the replayable model can be directly justified by the definition of a replayable. Furthermore, this model has been implemented in the form of Tatiana. Explanation of the notion of replayables and what can be done with them is the central aspect of instructing novice Tatiana users into how Tatiana should be used. The orthogonality of the operations means that when new kinds of operations are defined (e.g. a new transformation, visualisation or enrichment), the remaining operations are still applicable on the input or the output of the new operation. This makes it possible for Tatiana to be an extensible environment for which plugins can be created (and for which the return on investment of creating this plugin is benefiting from the other operations).

Because of this, if further work on understanding how analysis is performed expands on the notion of a replayable, this expansion will directly affect the replayable model, showing which aspects of it should be changed to make room for improvements on our work. Definition of the replayable model has initiated this development by looking at the kinds of replayables, visualisations and transformations which exist.

While we have noticed a slight overlap between transformation and enrichment (cf. §3.3), as afforded by the model, these operations are different in their analytic intention, the first serving to record the existence of events which can be observed in the trace file and to describe these events with properties which are as objective as possible, the second serving to

record additional knowledge as described by a given analyst and which can be considered in a hermeneutic fashion by another analyst. There is also a distinction to be drawn between automatic calculation of properties (mostly done by transformation) and the manual addition of properties (mostly done by enrichment). Finally, and more pragmatically, transformation produces results which only exist in the newly created replayable and any replayables further down that branch of the transformation tree; enrichment produces results which can also be propagated back up the transformation tree.

Another operation, the creation of collections, could be performed in two ways. The first would be to manually fill a new replayable with events from an existing replayable (all events in the resulting collection could then potentially be enriched with a common keyword). The other way would be to first enrich all these events with a common keyword and then apply a transformation which results only in events having the common keyword. Perhaps this simply illustrates that, due to the orthogonality between the two, transformation and enrichment can be performed in either order and as a consequence, the difference between these two ways of creating collections may be of little significance.

This work on visualisations, transformations and enrichments could be integrated back into the definition of replayables and of how analysis could be performed on them by looking more closely at *why* certain replayables and replayable visualisations are more appropriate than others to answer specific challenges.

5.3.1.3 Can the analytic challenges presented in our case studies be resolved through the artefact as defined?

While our case studies are in no way an exhaustive coverage of epistemologies and methodologies used in analysis of process data, they do cover a wide range, from ethnology to CSCL, from quantitative to qualitative, manual and automatic, and on corpora both with and without digital traces and with and without video traces. While Tatiana itself does not yet implement all the necessary visualisations, transformations and enrichments which were required, they could all be done without having to alter the model behind Tatiana, the definition of replayables or the set of operations which can be performed. This shows that many kinds of analysis can indeed be performed and explained using the subset of analytic artefacts defined as replayables.

5.3.2 The model for representing replayables

In chapter 1, we presented a type-model for describing how replayables can be represented and prescribing how a software environment for evaluating replayables should be built. We further examined this model to describe how the operations defined on replayables could be operationalised and what kinds of visualisations and transformations exist.

5.3.2.1 Does this model afford the operations defined on replayables?

By definition, the answer to this question is yes. The only qualificative which need be added concerns the representation of relationships which are inherent to the trace (e.g. the reply structure of a trace). As it stands, this structure cannot be recorded in a replayable alone in another way than to include it somehow in the replayable events (e.g. by having a "parent id" facet). To make this structure explicit, the information must be factored out of the replayable and into an enrichment. This goes against the general notion that enrichment should be information contributed by the researcher. One potential solution to this might be to consider all facets as contributed to events by enrichment relationships, drawing only the distinction between relationships which have been factored out from the original data and those which have been created by the researcher. This would make replayables even more like the graph structures proposed by ABSTRACT and NXT. Doing so, however, would make it more difficult to implement transformations as they would have to take the entirety of the enrichments of a corpus into account.

Another question which might be asked is whether the possibility of creating N-ary relationships is wise. Indeed, relationships which are attached to one event only behave differently from those which link two events. We have not yet found a case where it is useful for N to be greater than two, although it is conceivable that an event might, for example, be contingent on two other events considered together.

5.3.2.2 Can this model inform the implementation of an environment for analysis through the manipulation of replayables?

The existence of Tatiana and the fact that the aspects of Tatiana which are described by this model match the model very closely are indicative that this model can be used to implement a replayable manipulation environment. Slight discrepancies between Tatiana and the model variously illustrate the model's shortcomings and the fact that Tatiana still does not implement certain aspects.

Aspects not yet implemented by Tatiana include a system for verifying that transformations will execute correctly (correct inputs as specified by the transformation), and details of facet representations (that facet values can be a set of values and that certain kinds of facet representation are more or less appropriate to certain visualisation and transformation purposes). Although we have suggested that when events are grouped together (creating a representative of a new observed event, implemented as a set of anchors), enrichment facets on that new observed event might be automatically calculated from the enrichment facets on its "children" (cf. §3.5), we have not yet encountered a situation where this would be of practical use (i.e. we have only encountered cases of wrapping, where we wish to represent that a new event should be cover the time slice represented by its members and should not "inherit" any of its members' facets) – for this reason we have neither explored the semantics in detail, nor implemented such a feature. Furthermore, enrichments are not yet automatically expanded into replayables for transformation purposes. This point illustrates that Tatiana's file-based implementation might not be ideal (i.e. certain database features such as views and triggers are missing when considering files).

The shortcomings of the model described in the previous section are also found in the difficulty of implementation. Relationships provided by enrichments currently only provide a single facet and Tatiana makes an explicit difference between unary and binary relationships, particularly when it comes to visualisation.

Finally, the orthogonality of the different operations works well in theory and in the model. In practice, however, distinguishing visualisation and transformation, transformation and enrichment and enrichment and visualisation can prove problematic.

Particularly when transformations produce results which are not replayables, or when a visualisation depends on particular information which is not present in all replayables (e.g. the

presence of a certain facet), these transformations might sometimes best be implemented by a new visualisation or enrichment plugin. For example, implementing the state-transition graphs which are described in Hilbert & Redmiles (*2000)* could be done with a visualisation plugin. Implementing a threaded discussion visualisation plugin might involve requiring that replayables which could be visualised have a "parent id" facet. This facet could be contributed by a transformation. Another way would be for users to provide a rule which allows the plugin to "calculate" the children. This is only a minor problem as the intention is clear: transformations describe content and visualisations describe form. In other words, this does not imply that the model is in some way poorly adapted (the distinction between content and presentation is common – e.g. html and css, models and views, etc.). Pragmatically however, it may be simpler for visualisation plugins to do more.

We have already discussed the relationship between transformation and enrichment (cf. 5.3.1.2). This problem is exacerbated by the current absence of a means to automatically factor out enrichments from replayables. However, this problem does not reflect on the model, but on wider aspects of defining replayables and the difference between the two operations.

Our desire to provide automated transformations which update automatically has led us to distinguish editable and non-editable replayables. It remains to be seen whether replayables which are created from a transformation can be made editable and preserve a coherent semantics.

Finally, as we have seen in chapter 1, we have chosen to make enrichments visualisable and to include these visualisations in the synchronisation mechanism. The pragmatic reason for this, is that many enrichments are intimately linked to certain forms of visualisation (specifically, the aspect of visualisation which makes it possible to edit the enrichment). It is easier to just implement a single plugin rather than two, the first for storing enrichments and the second for editing them. If we were to consider binary enrichment and unary enrichments as separate, this might reduce this problem. This would not mitigate the fact that enrichments are designed to potentially span across multiple replayables.

5.3.2.3 Could all the tools presented in the state of the art be implemented using this model?

One of the major motivations for the work presented in this dissertation is that there is no coherent framework for describing the various analysis tools which exist and the artefacts

which they create. For this model to be properly generic and be a reusable source of information for the trace engineering community, it is important to show that this model can serve as a basis for reasoning about analysis in general, not just in the way it is implemented in Tatiana. The state of the art we have described attempts to paint a broad picture of current analysis possibilities and is just the tip of the iceberg with regard to existing tools. However, it is reasonable to assume that, if this model can be used to describe all the tools which exist in the state of the art, it can be used to describe almost any kind of analysis software and the analysis practices afforded by this software. Furthermore, this would mean that all such analysis tools could be implemented as Tatiana plugins, providing a powerful means of sharing corpora and analysis methods.

The very fact that this model describes replayables and that replayables are abstracted from the state of the art is a first indication of this generality. While it would be tedious and difficult (for reasons of conflicting vocabularies rather than conflicting concepts[23]) to show, for each tool and model in the state of the art, that all objects which can be represented can be represented by our replayable model, we show some examples to illustrate the broadness of the replayable model.

As shown in §3.3.2 replayables cover all the notions presented in M-traces (aside from the explicit modelling aspect). From this, it follows that ABSTRACT can also be described using the replayable model (and the existing scoresheet visualisation plugin can re-create the visualisations of ABSTRACT). The video analysis tools whose underlying structure is generalised by NXT present a slightly greater challenge due to the vocabulary used. Each annotation tree (cf. §1.4.3) could be represented by a replayable, with each annotation object being an event. The parent-child relationships described in annotation trees would use the fact that observed event can subsume other observed events. The other relationships existing in annotation trees would be represented by enrichment relationships.

The various trace-analysis tools all deal with events with various properties and different visualisations of these events, which are afforded by our model. All the functionalities present in ESDA tools such as MacShapa can also be represented with the replayable model. Each MacShapa data stream (cf. §1.4.5.2) would become a series of events, with a set of data streams being grouped together in a replayable. The only limitations come from the analysis and extraction of patterns, which do not exist in Tatiana or in the replayable model. While the

[23] But also because we have not described our model with a very precise formalism, nor are most of the tools presented in the state of the art described with sufficient precision that we can prove that we can represent all the artefacts they can create.

results of such extractions are not replayables, they are time-oriented and perhaps the next logical step in analysing activity is understanding how we can use and interpret such patterns in the context of analysis (e.g. in the case study presented in §5.2.5).

Contingency graphs would also be easy to represent as replayables, with vertices and edges translating naturally into events and relationships. Furthermore, different kinds of contingencies could be stored in different enrichments allowing them to be applied and removed from the graph as needed, much like map transparencies can be added and removed from a map.

Thus, we argue that all software and artefacts presented in the state of the art *could* use replayables as a data model and replayable operations as basic operations. As such the replayable model we describe would be a good place to start for future work in the trace engineering field.

5.3.3 A software environment for manipulating replayables

While we do not intend to evaluate the software itself (which exists so that the implementability of the replayable model and the validity of the replayable concept for analysis can be evaluated), we briefly examine the outstanding problems in Tatiana, mainly with regard to the future goal of analysing a widespread usage of Tatiana in order to better understand research practices.

A common obstacle to any software's adoption is usability. The usability of Tatiana has evolved from "impossible to use" to "barely passable" and still leaves a number of gaping holes. In particular, the notion that all analytic moves can be expressed by some combination of the four features of Tatiana is powerful but requires much better explanation and integration with existing practices. Search and filter, for example, are common functionalities in other tools. In Tatiana these are both performed through transformations (using the poorly worded "Tatiana filter"). A better integrated user interface would propose these as special operations, transparently hiding the fact that they use transformation from the user. Other common software features such as "print", "copy" and "paste" are conspicuously absent from Tatiana.

In other cases, some functionalities afforded by Tatiana are hard to use. The rules for scoresheet visualisations in particular take a long time to create (from a user-interface point of

view) and are hard to understand (from a cognitive model point of view). Filter inputs can be difficult to choose unless they are well documented. For example, the generic transformation for selecting only events where a certain facet has a certain value demands three inputs: a replayable, a facet name and a facet value. None of these terms are understandable to a novice user.

Tatiana is not yet complete – in the sense that it does not fully implement the replayable model. This and a lack of documentation are still strong barriers to adoption by programmers who, rather than write new analysis tools, could implement them as Tatiana plugins and by users who may find their first experience with Tatiana frustrating and not consider the investment of learning how to use the full power of Tatiana worthwhile.

5.3.4 Global evaluation

Overall, our evaluation (both from our own criteria for evaluation as researchers within the field of trace engineering and from the point of view of our users – researchers who study interactions) is positive and provides clear paths for improvements and areas which need further study. Aside from the evaluation criteria for individual aspects of the thesis laid out at the beginning of this section, we can also examine our results globally with regard to the various challenges identified in particular in §1.2.2 and §2.2.

Synchronisation and navigating up and down among levels of analysis is something we have solved (like many before us – although the creation of separate artefacts related by synchronisation is innovative in its intention if not in the end result[24]). Hopefully Tatiana will assist researchers attempting to solve the methodological and epistemological challenges posed by multi-level analysis. In the same way, a customisable tool such as Tatiana may assist in the difficulty in handling such large levels of data and scaling up analysis.

The problem of capitalising on analyses and sharing analyses and corpora is a core theme of this dissertation and which we have addressed successfully – within the limits of our understanding of analysis. We have shown some of the capitalisation possibilities afforded by reified analyses. Further work needs to be done to understand other analytic artefacts and to make Tatiana better able to handle and share complete corpora or subsets of such corpora. In a similar vein, documents relating the context of a study (as identified in the MULCE project)

[24] Meaning that we explicitly want different events at different levels in different artefacts to be related through synchronisation – as opposed to some other more specific mechanism, such a structure. This does not seem to be the case in other tools.

need to be integrated into our model, affording a better integration of the context which is not exhibited by interactions.

In §1.4.3 we reported that Dybkjaer *et al.* (2001) called for the following requirements for video analysis tools

- Flexible and open architecture;
- Separation of the user interface from the internal application logic;
- Transcription support;
- Annotation support at different levels;
- Powerful functions for query, statistical analysis and information extraction;
- Synchronised visualisation;
- Easy to use interface;
- Support for addition of new coding schemes and for defining new visualisations;
- Possibility of importing existing data sources.

All of these are present in Tatiana.

5.4 Perspectives

As the above evaluation shows, we have satisfied the majority of the goals set out for this dissertation. The identified outstanding issues provide guidance on what issues are appropriate for further study and how our work can serve as a basis for this study. In this section, we discuss perspectives from two points of view within the trace engineering field: that of better understanding and supporting analysis practices and that of possibilities for automated treatment and analysis of traces.

5.4.1 Understanding and improving analysis practices

Our work has dealt with objects we have termed replayables. With the help of tools such as Tatiana, we hope to gain a better understanding of how analysis can be performed with replayables and what the limitations of such objects are. This will allow us to improve

Tatiana, document existing analysis methodologies and determine how other kinds of analytic artefacts (such as aggregations and static documents) can be integrated into our work.

In particular, our work allows the reification of analyses such as categorisations and contingency graphs. The work begun in Tatiana will allow us to create, explore and automatically evaluate such data, helping to address some of the key issues which have been identified, particularly in CSCL, such as moving beyond adjacency pair analysis, scaling up analysis both to multiple levels of analysis and larger datasets, discovering patterns and pinpointing pivotal moments in interaction.

Replayables are artefacts which are destined to assist a researcher in gaining a deeper understanding of their data through an iterative, interactive, exploratory practice. When using such objects to succinctly present results to other researchers (i.e. creating *products* as defined in §2.2) we have consistently had to annotate and describe them in order for them to communicate the essence of the knowledge they embody. Understanding the difference between the cognitive affordances of analytic artefacts which assist the discovery of new knowledge and products which are designed to communicate that knowledge seems like a key area for further research.

In turn, this work feeds back into the use of traces for other users than researchers, enabling the use of traces as feedback both to users and teachers, and as a means to render computer environments adaptable, providing personalised experiences to their users.

5.4.2 Sharing corpora and analyses

As the title of this dissertation suggests, one of the motivations for our work is to capitalise on analyses. This not only includes representing them in a way which enables further manipulation, but also in a way that is shareable. This is something which we have achieved with Tatiana and which will contribute to the ongoing work on sharing corpora.

Sharing corpora and analyses is important for several reasons. As we saw in the case study on collaborative design in engineering, collaboration between researchers in different fields on the analysis of a given corpus can lead to the emergence of new interpretations. Being able to share both corpora and analyses will also allow researchers to better understand and evaluate each others' work and findings.

Developing the notion of replayables also provides a vocabulary for describing and justifying analysis methodologies, showing the extent of "objectivity" contained in a given result and

allowing the choice of an analysis methodology to achieve a balance between epistemology (whether the methodology is acceptable from a scientific standpoint) and economy (whether the methodology is cost effective). This also provides a vocabulary for discussion and sharing of analysis protocols, allowing researchers to improve explanation, discussion and replication of methodologies.

Many authors call for interoperability between tools. As we have shown, interoperability between plugins within an analysis environment avoids the necessity of replicating existing functionality. For this reason, further exploration of the orthogonality of analytic moves will enable new kinds of moves to be developed which integrate with existing orthogonal moves, affording large numbers of new analysis (through the combination of various operations) rather than simply providing an "extra" analysis method. We have noted, however, that our "competitors" who do this find it much easier to explain what their tool does (as it only does one thing) and their users, in turn, find it much easier to ascertain whether the tool is something they can use or not. Some might see this as the balance between a Swiss army knife (which does many things poorly) and a specialised kitchen knife (which does one thing exceedingly well). In the development of software however, we believe it is possible to create powerful multi-purpose tools without sacrificing specificity.

5.4.3 Describing the knowledge embodied in traces

Because our work is grounded in the notion of flexibility and letting the analyst provide the meaning of various events and their facets (e.g. a social network analysis plugin could say "I have found facets X, Y and Z. Which one represents the user?"), we deliberately avoided the notion of creating ontologies to describe replayables. In other words, while our replayables conforming to an implicit model, we do not make it explicit because the cost to the user would outweigh the benefits – in other words, we have provided a syntax but not a means for defining semantics on that syntax, which currently only exist in the "mind" and practices of the researcher. However, as the understanding of traces improves, we will be able to describe certain classes of traces and events and the methods which can be used to transform, visualise and enrich them. This understanding can be linked with understanding of domains of discussion (e.g. a corpus wherein engineering is discussed) and of domains of analysis (e.g. the notion of utterance, topic, reply) to provide (as much as possible) automated analysis of

corpora through all automated methods which are detected as appropriate such as semantic, structural and social analysis.

As work on ontology-matching improves, we can imagine situations where the structure of a trace would enable the detection of the kind of actions embodied in that trace and allow analysis ontologies (e.g. social network analysis knows about users and events which exhibit relationships between users) to be matched up with trace ontologies. This could lead to new applications of existing techniques. For example, the similarity in properties between users and concepts, both of which have identifying representations, could be used to conduct "social network analysis" on concepts, showing which concepts are central and which concepts are peripheral to a discussion.

This work could also lead to analysis tools which know how to interoperate with the software they analyse, providing various forms of immediate feedback for consumption by the software itself or for visualisation by end users.

5.4.4 Methods for trace analysis

Automated analysis and computer-support for analysis would also benefit from explicitly defined models of traces. In particular, we see important benefits to be gained in computer science techniques which use information retrieval to locate data which conforms to a pattern (when researchers know what they are looking for) and data mining to identify candidate patterns (when researchers do not yet know what they are looking for). These techniques will be all the more promising if they are adapted to analysis of replayables and their enrichments.

As colleagues in our lab work on information retrieval, we have done some exploratory work to address this topic (Dyke, Beigbeder, Girardot, & Lund, 2009). Information retrieval techniques are majoritarily based on the notion of matching a query against a series of documents, with a distance metric being used to compare queries and documents with each other. We translate this notion to query replayables by determining that a document (what should be returned by a query) is a range of events in a replayable which matches the query (e.g. searching for "hydraulic cable" might find a set of events where one user talks about the cables and another refers to the hydraulic one). This poses interesting indexing problems as documents are not fixed but can span any number of events.

In a similar way, other computer science methods can be adapted to the particular problems posed by replayables and analysis, affording both new analysis techniques to the field of

interaction analysis and new applicative domains to computer science. In particular the analysis of enrichments should provide new ways for integrating manual and automatic analysis. For example, a structure made explicit by an analyst through a graph analysis could then be capitalised upon through automated analysis and visualisation which, in turn could feed further manual analysis.

We see Tatiana as an environment within which new automated techniques can be implemented and tested in combination with Tatiana's existing functionalities, providing a way for such techniques to be accessible to researchers who want to use these techniques without having to pioneer or develop them themselves (assuming adequate documentation is provided).

5.5 Conclusion on results and perspectives

In this chapter, we presented the case studies in which Tatiana was used to evaluate the concepts it implements. These case studies helped provide many of the requirements on which the work presented in this dissertation is based and illustrate the soundness of using the concept of replayables to analyse human interactions. They cover a wide range of data kinds (chat, video, forum, blog), analysis methodologies (quantitative, qualitative and various mixed methods) and research fields (CSCL, ethnography, collaborative design). In all cases, Tatiana, in its current state and with the development of appropriate plugins, *could have been used* to solve the analytic challenges faced by researchers. While in many cases, Tatiana was not successfully used, this was due to the state of development of Tatiana at the time or to the lack of appropriate plugins – and occasionally to poor usability. While this is of little consolation to the researchers with whom we have been collaborating, it is an encouraging result regarding our work as these are problems with Tatiana itself and *not* with the underlying concepts. We are now entering a phase where we concentrate on improving the usability of Tatiana and its alignment with these underlying concepts.

We then evaluated both the notion of a replayable as an analytic artefact designed to answer certain analytic problems and the model we developed for representing replayables. We concluded that, within the discussed restrictions (e.g. that replayables can only be used for certain kinds of analysis) replayables represent all the analytic artefacts and moves found both

in the state of the art and our case studies. Furthermore, their definition in terms of four core operations affords a reasonably easy transition to a model which can be reused by researchers in the field of trace engineering and adapted as our understanding of such analytic artefacts improves. The replayable model itself was constructed to afford the operations defined on replayables and served to inform the implementation of Tatiana. We discussed the problems encountered when implementing this model. We concluded by explaining how this replayable model could serve as a basis for implementing the tools present in the state of the art and, therefore, as a means to discuss various methods in trace engineering, abstracted from individual tools. Finally, we gave a brief overview of the problems with Tatiana itself, particularly those which are an obstacle to widespread adoption.

We concluded this chapter by examining the perspectives which are opened up by our work in terms of improving our understanding of analysis, our ability to share corpora and analyses, the means by which the knowledge embodied in traces could be exploited and the kinds of automatic analysis which could be performed in the future.

Conclusion

In this dissertation, we addressed the problem of analysis of human interactions, particularly in the field of CSCL. Specifically, we attempted to provide better tools to analysts who wanted support in creating the artefacts which embody their analysis. Our work lays the foundations for analyses to be artefacts which are not only produced, but can also be modified and re-used in powerful ways.

The need for providing such tools stemmed from a long-standing collaboration between computer scientists and researchers who analyse interaction (and in particular collaborative learning) from a socio-cognitive perspective. The concrete context of the LEAD project provided our main use case: how can situations where students who are interacting through a blend of face to face and computer-mediated activity be recorded and analysed without losing too much of the context? For this reason, while addressing the analysis of collaborative interactions in general, we focussed on CSCL interactions in particular.

We conducted this work from the perspective of *trace engineering*, which we see as the science which expands and records our knowledge of the ways for modelling, representing and exploiting traces for a variety of purposes. As such, understanding the purposes (such as analysis) and the constraints which such purposes impose on our engineering is a key focus, both for providing adequate solutions and for creating a reusable body of knowledge about these purposes and how they relate to our engineering choices.

We first gave an overview of the field of CSCL and some of the difficulties faced by analysts attempting to understand CSCL data. We defined a *trace* as being a recording of the observable events of an activity which are considered pertinent with regard to some goal. We then examined the state of the art for the modelling of traces, for analysis of traces and of tools which assist analysis. This led us to identify three issues: we did not have enough knowledge of analysis in general or of CSCL in particular to proceed; the existing models for the representations of traces and analyses were not framed in such a way that their suitability for modelling CSCL analysis could be immediately ascertained and the existing tools most often presented a list of features which, while clearly fulfilling the requirements of the tool users, were not presented in way which enabled us to understand how features should be integrated, or which features were appropriate for which kinds of analysis problems.

To address these issues, we set out three goals for our work: to further our understanding of analysis and of the class of analytic problems we would address by describing a generic analytic artefact through the operations which can be performed on it; to model this artefact in such a way as to enable the operations defined on it; to implement an environment for creating, editing and reusing artefacts of this type in order to evaluate the previous two results through case studies. We concluded our definition of the problem by laying out the criteria by which we would be evaluating our results.

In order to gain a better understanding of analysis and define the scope of the analytic problems our model would address, we extended our state of the art to discussions of analysis within a wider context. We described analysis as a process wherein analysts iteratively create analytic artefacts based on their original data until these artefacts represent a body of knowledge which can lead to interpretation and publication of results. We then examined the specificities of CSCL traces and analysis. We concluded that the field of CSCL combines the problems posed by analysis of learning, analysis of interaction and analysis of multimodal data thus compounding the difficulty but not posing any additional specific challenges. We proposed to address the range of analytic problems faced by the analysis of human interaction activities.

In light of this, we re-examined the artefacts created during analysis (as evidenced by the tools which support the creation of such artefacts) and described their features, abstracting out a generic artefact which we called a *replayable*. Replayables can be synchronised with each other, can be visualised in a variety of ways, can be created and transformed, both automatically or by hand, and can be enriched with additional data during analysis.

Having defined the notion of a replayable and its scope of applicability for analysis, we addressed the issue of modelling replayables. We first examined what was entailed by a model, showing how creating a descriptive type-model of existing replayables would provide us with a prescriptive model for the description of a software environment which, in turn, would enable us to create token-models of these replayables within the software environment. In the model we proposed, traces are a special form of replayable which is made up of observed events. Non-trace replayables are made up of events which are proxies for the observed events. Events encapsulate a timestamp, a designation of the event they represent and a set of named and typed facets which can be either inherent to the event, or provided via the relations which enrich the observed event they represent. We then examined certain types

of facets and events more closely along with their consequences for visualisation and transformation.

We presented the tradeoffs involved in creating specialisations of a general replayable model which would make it possible to describe certain classes of replayable more precisely. We concluded that defining such a model so that the advantages would outweigh the benefits was beyond the scope of our work, as such a system must allow both extendable types, inheritance and type discovery/calculations. Without such features, it would be impossible to create generic transformations and visualisations and it would be very cumbersome for the analyst to have to explicitly define such models for every replayable they wish to create. By leaving the semantics of replayables implicit, however, it is possible to exploit analysts' inherent knowledge of the data they are analysing.

This model was implemented in the Tatiana analysis environment. Tatiana provides a generic means for using replayables for analysis, incorporating visualisations, transformations, enrichment and ubiquitous synchronisation. The environment is based on the Eclipse Rich Client Platform and allows new visualisation and enrichment types to be added with a plugin mechanism. New transformations can be created by dropping the XQuery file for that transformation in the correct folder. Tatiana can also be synchronised with external replayers which implement the synchronisation protocol.

Tatiana was used in ten major case studies. While these case studies highlighted Tatiana's usability shortcomings (some of which were handled and some of which still exist), the notion of replayable and the replayable model presented in this dissertation seem adapted to solving all the analytic problems provided by the case studies. The other criteria for evaluating our work were also met with satisfaction: the notion of a replayable and its model are generic enough to represent the artefacts found in the state of the art; furthermore, the three levels of definition, model and implementation are related in such a way that the links between these levels are clear and easy to understand – and modify.

In this work, we create an integrated and consistent framework within which certain specific analytic artefacts can be created and exploited by analysts. We clearly defined the rationale for the existence of such objects and showed how the definition informs the model and the model informs the implementation. In this way, our work can be reused at all levels.

Analysts can use Tatiana as it is to analyse many situations of human interaction. Programmers can extend Tatiana to incorporate new kinds of visualisation, enrichment and transformation. Researchers who want to provide tools for automated analysis can use the

replayable model we provide as a basis from which to work. Finally, if the analytic artefacts which are not replayables are described in relationship to replayables, it should be possible to adapt and improve the model and implementation of such analytic artefacts in a consistent way.

Further work in trace engineering should examine how the knowledge embodied in traces (the activity), in the context (the applicative domain) and in the analysis (the researcher's knowledge) can be represented and modelled. This will make it possible to maximise the automatic discovery of phenomena which are of interest to researchers. Although automatic analysis will never replace the work of human researchers, it can help them create artefacts which represent analytic knowledge and explore large quantities of data within several interrelated artefacts to pinpoint areas of interest for more detailed manual analysis.

As this work enables analyses to be modelled and reused, they can also be shared, laying the technological foundations for a social change in the way research is conducted within the CSCL community. As more and more researchers choose to share not only their data, but also their analyses of that data, we will need a better understanding of the nature of these artefacts, moving from immutable products to documents which can be edited, reused and integrated with each other in order to fuel scientific criticism and advancement.

References

Anderson, J. C., Slater, N., & Lehnardt, J. (2009). *CouchDB: The Definitive Guide*. O'Reilly.

Amelsvoort, M.A.A., Andriessen, J.E.B., & Kanselaar, G. (2008). How students structure and relate argumentative knowledge. *Computers in human behavior*, 24, 1293-1313.

Avouris, N., Fiotakis, G., Kahrimanis, G., Margaritis, M., & Komis, V. (2007). Beyond logging of fingertip actions: analysis of collaborative learning using multiple sources of data. *Journal of Interactive Learning Research*, *18*(2), 231.

Bachimont, B. (2004). Pourquoi n'y at-il pas d'expérience en ingénierie des connaissances. In *Actes du colloque IC'2004*. Lyon.

Badre, A. N., Guzdial, M., Hudson, S. E., & Santos, P. J. (1995). A user interface evaluation environment using synchronized video, visualizations and event trace data. *Software Quality Journal*, *4*(2), 101–113.

Baker, M. J., Andriessen, J., Lund, K., van Amelsvoort, M., & M, Q. (2007). Rainbow: a framework for analysing computer-mediated pedagogical debates. *International Journal of Computer-Supported Collaborative Learning*, *2*, 315–357.

Baker, M., Andriessen, J., & Lund, K. (2009). Socio-relational, affective and cognitive dimensions of CSCL interactions: integrating theoretical-methodologial perspectives. In *Symposium held at CSCL09*. Rhodes, Greece.

Becker, R. A., & Cleveland, W. S. (1987). Brushing Scatterplots. *Technometrics*, *29*, 127–142.

Bernard, F.-X., & Baker, M. (2009). Une analyse des processus d'appropriation d'un environnement informatique pour l'apprentissage collaboratif dans la classe. In *Actes du colloque EIAH 2009*.

Bernsen, N. O., Dybkjaer, L., & Kolodnytsky, M. (2003). An interface for annotating natural interactivity. *Current and New Directions in Discourse and Dialogue*, 35.

Bertelsen, O. W., & Bodker, S. (2003). Activity theory. *HCI models, theories, and frameworks: Toward a multidisciplinary science*, 291–324.

Blake, C. T., & Rapanotti, L. (2001). Mapping Interactions in a Computer Conferencing Environment. In P. Dillenbourg, A. Eurlings, & K. Hakkarinen (Eds.), *Proceedings of the European Perspectives on Computer Supported collaborative Learning Conference, Euro-CSCL 2001*. Maastricht.

Card, S., Mackinlay, J., & Shneiderman, B. (Eds.). (1999). *Readings in Information Visualization - Using Vision to Think*. Morgan Kaufmann.

Carletta, J., Evert, S., Heid, U., Kilgour, J., Robertson, J., & Voormann, H. (2003). The NITE XML Toolkit: flexible annotation for multimodal language data. *Behavior Research Methods Instruments and Computers*, *35*(3), 353–363.

Carletta, J., Kilgour, J., O'Donnell, T., Evert, S., & Voormann, H. (2003). The NITE object model library for handling structured linguistic annotation on multimodal data sets. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*.

Cassier, J.-L. (in preparation). *Modélisation de l'Activité de Conception Collaborative Synchrone de Produits Industriels.* Thèse présentée à l'Université de Lyon.

Champin, P. A., Prie, Y., & Mille, A. (2004). MUSETTE: a framework for knowledge capture from experience. In *12eme Atelier du Raisonnement a Partir de Cas (RaPC'04)* (pp. 85–97). Villetaneuse, France.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., et al. (2006). Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*.

Choquet, C., & Iksal, S. (2007). Modeling Tracks for the Model Driven Re-engineering of a TEL System. *Journal of Interactive Learning Research*, *18*(2), 161–184.

Choquet, C., & Iksal, S. (2007). Modélisation et construction de traces d'utilisation d'une activité d'apprentissage : une approche langage pour la réingénierie d'un EIAH. *Revue Sticef*, *14*.

Ciekanski, M., & Chanier, T. (2008). Developing online multimodal verbal communication to enhance the writing process in an audio-graphic conferencing environment. *Recall*, *20*(2), 162–182.

Corbel, A., Jaillon, P., Serpaggi, X., Baker, M., Quignard, M., & Lund, K. (2003). DREW : Un outil Internet pour créer des situations d'apprentissage coopérants. In Desmoulins, Marquet, & Bouhineau (Eds.), *Actes de la conférence EIAH 2003* (pp. 109–113). Strasbourg.

Cox, R. (2007). Technology-enhanced research: educational ICT systems as research instruments. *Technology, Pedagogy and Education*, *16*(3), 337–356.

De Chiara, R., Di Matteo, A., Manno, I., & Scarano, V. (2007). CoFFEE: Cooperative face2face educational environment. In *CollaborateCom 2007. International Conference on Collaborative Computing: Networking, Applications and Worksharing* (pp. 243–252).

De Laat, M. F., M, C., & Wegerif, R. (2008). Facilitate the Facilitator: Awareness Tools to Support the Moderator to Facilitate Online Discussions for Networked Learning. In *Proceedings of the 6th International Conference on Networked Learning* (pp. 80–86). Halkidiki, Greece.

Denning, D. E. (1986). An Intrusion Detection Model. In *Proceedings of the Seventh IEEE Symposium on Security and Privacy* (pp. 119–131).

van Diggelen, W., Jansen, J., & Overdijk, M. (2008). Analysing and presenting interaction data: a teacher, student and researcher perspective. In *Proceedings of ICLS 2008*. Utrecht, The Netherlands.

Dillenbourg, P. (1999). What do you mean by "collaborative learning". In P. Dillenbourg (Ed.), *Collaborative learning: cognitive and computational approaches.* Amsterdam, Netherlands: Elsevier.

Dillenbourg, P. (2005). Designing biases that augment socio-cognitive interactions. In R. Bromme, F. W. Hesse, & H. Spada (Eds.), *Barriers and biases in computer-mediated knowledge communication – And how they may be overcome* (pp. 243–264). Dordrecht, Netherlands: Kluwer.

Djouad, T. (2008). Analyser l'activité d'apprentissage collaboratif: Une approche par transformations spécialisées de traces d'interactions. In *2ème rencontre des jeunes chercheurs RJCEIAH08* (pp. 93–98).

Dybkjaer, L., Berman, S., Kipp, M., Olsen, M. W., Pirelli, V., Reithinger, N., & Soria, C. (2001). *Survey of Existing Tools, Standards and User Needs for Annotation of Natural Interaction and Multimodal Data*. ISLE Natural Interactivity and Multimodality Working Group Deliverable D11.1.

Dyke, G., Beigbeder, M., Lund, K., & Girardot, J.-J. (2009). Les traces d'interactions humaines, un nouveau domain d'application pour la RI. In *Actes de la sixième conférence francophone en Recherche d'Information et Applications* (pp. 397–408). Hyères, France.

Dyke, G., & Lund, K. (2007). Implications d'un modèle de coopération pour la conception d'outils collaboratifs. In *Actes d'EPAL 2007*. Grenoble, France.

Eclipse (n.d.). Retrieved from http://www.eclipse.org.

Educational Data Mining (n.d). Retrieved from http://www.educationaldatamining.org/.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data (Rev. ed.)*. Cambridge, Ma: MIT Press.

Erkens, G., & Jansen, J. (2008). Automatic coding of communication in collaboration protocols. *ijCSCL*, *3*(4).

Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery: An Overview. In *Advances in Knowledge Discovery and Data Mining* (pp. 1-34). MIT press.

Fiotakis, G., Fidas, C., & Avouris, N. (2007). Comparative usability evaluation of web systems through ActivityLens. In *proc. PCI 2007*. Patras, Greece.

Fisher C., & Sanderson P. (1996). Exploratory sequential data analysis: exploring continuous observational data. *interactions*, *3*(2), 25–34. doi: http://doi.acm.org/10.1145/227181.227185.

France, L., Heraud, J.-M., Marty, J.-C., & Carron, T. (2007). Visualisation et régulation de l'activité des apprenants dans un EIAH tracé. In *Actes INRP/EIAH 2007*. Lausanne, Suisse.

Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New generation computing*, *9*(3), 365–385.

Gelmini Hornsby, G., Ainsworth, S., Buda, M., Crook, C., & O'Malley, C. (2008). Making your views known: the importance of anonymity before and after classroom debates. In *Proceedings of ICLS 2008*. Utrecht, The Netherlands.

Georgeon, O. (2008). *Analyse de traces d'activité pour la modélisation cognitive: application à la conduite automobile*. Thèse présentée à l'Université de Lyon.

Georgeon, O., Henning, M., Bellet, T., & Mille, A. (2007). Creating Cognitive Models from Activity Analysis: A Knowledge Engineering Approach to Car Driver Modeling. In *International Conference on Cognitive Modeling*. Ann Arbor, Michigan.

Goodman, B. A., Drury, J., Gaimari, R. D., Kurland, L., & Zarella, J. (2006). *Applying User Models to Improve Team Decision Making*. MITRE.

Giguet, E., Lucas, N., F.-M, B., & Bruillard, E. (2009). Share and explore discussion forum objects on the Calico website. In *Computer Supported Collaborative Learning Practices: CSCL2009 Conference Proceedings*. Rhodes, Greece.

Greenhalgh, C., French, A., Humble, J., & Tennent, P. (2007). Engineering a replay application based on RDF and OWL. In *Online Proceedings of the UK e-Science All Hands Meeting*. Nottingham.

Grésillon, A., & Lebrave, J.-L. (1983). Manuscrits-Ecriture. Production linguistique. *Langages*, *69*.

Harkavy, M. (Ed.). (1994). *Webster's new encyclopedic dictionary*. Black Dog and Leventhal publishers inc.

Harrer, A., Barros, B., Bollen, L., Dimitracopoulou, A., Fesakis, G., & Hullshoff, C. (2004). *Unified framework on interaction analysis*. ICALTS JEIRP.

Harrer, A., Hever, R., & Ziebarth, S. (2007). Empowering researchers to detect interaction patterns in e-collaboration. *Frontiers in Artificial Intelligence and Applications*, *158*, 503–510.

Harrer, A., Zeini, S., Kahrimanis, G., Avouris, N., Marcos, J. A., Martinez-Mones, A., et al. (2007). Towards a Flexible Model for Computer-based Analysis and Visualisation of Collaborative Learning Activities. In *Proceedings of CSCL 2007*. New Jersey, USA.

Harrer, A., Zeini, S., & Pinkwart, N. (2006). Evaluation of communication in web-supported learning communities–an analysis with triangulation research design. *International Journal of Web Based Communities*, *2*(4), 428–446.

Hilbert, & Redmiles. (2000). Extracting usability information from user interface events. *ACM Comput. Surv.*, *32*(4), 384–421. doi: http://doi.acm.org/10.1145/371578.371593.

Hmelo-Silver, C. E., Chernobilsky, E., & Jordan, R. (2008). Understanding collaborative learning processes in new learning environments. *Instructional Science*, *36*(5), 409–430.

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science: A Multidisciplinary Journal*, *19*(3), 265–288.

IMS-LD (2003). Instruction Management System, Learning Design Specification Version 1, final specification. Retrieved from www.imsglobal.org/learningdesign/index.html.

Jermann, P., & Dillenbourg, P. (2003). Elaborating new arguments through a CSCL script. In *Arguing to learn: confronting cognitions in computer-supported collaborative learning environments* (pp. 205–226). Dordrecht, Netherlands: Kluwer.

Jermann, P., Soller, A., & Mühlenbrock, M. (2001). From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. In *Proceedings of the first european conference on Computer-Supported Collaborative Learning* (pp. 324–331). Maastricht, the Netherlands.

JSON (2009). Retrieved from http://www.json.org/.

Kahrimanis, G., Papasalouros, A., Avouris, N., & Retalis, S. (2006). A Model for Interoperability in Computer Supported Collaborative Learning. In *Proc. ICALT 2006 - The 6th IEEE International Conference on Advanced Learning Technologies* (pp. 51–55). Kerkrade, Netherlands.

Kapur, M., & Kinzer, C. K. (2009). Productive failure in CSCL groups. *International Journal of Computer-Supported Collaborative Learning*, *4*(1), 21–46.

Koschmann, T. (2002). Dewey's contribution to the foundations of CSCL research. In G. Stahl (Ed.), *Proceedings of CSCL 2002: foundations for a CSCL community* (pp. 17–22). Mahwah, NJ: Laurence Erlbaum Associates.

Kühne, T. (2006). Matters of (meta-) modeling. *Software and Systems Modeling*, *5*(4), 369–385.

Laflaquière, & Prié, Y. (2009). L'expérience tracée des activités conjointes instrumentées. In *Atelier ICT 2009*. Caen, France.

Laflaquière, J., Champin, P. A., Prié, Y., & Mille, A. (2005). Approche de modélisation de l'expérience : utilisation de systèmes complexes pour l'assistance aux tâches de veille informatiquement médiées. In A. David (Ed.), *ISKO-France 2005* (pp. 209–230).

Vandoeuvre-lès-Nancy, France: Presse universitaire de nancy.

Lalanne, D., & Ingold, R. (2004). Documents statiques et multimodalité. L'alignement temporel pour structurer des archives multimédia de réunions. *Temps et Documents*, *8*(4), 65–89.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*, 259–284.

Law, N., Lu, J., Leng, J., Yuen, J., & Lai, M. (2009). Exploring Knowledge Building Discourse using argumentation markers. In *Workshop on Analysis of computer-supported collaborative learning (CSCL) discourse: advances in theory and analysis tools, CITE Research Symposium 2009*.

LEAD (2006). *Problem Solving through face to face networked interaction in the classroom*.

Lebrave, J.-L. (1989). Les proto-termes dans les variantes d'écriture. *DRLAV ("Signes et sens")*, *40*, 89–113.

Lomborg, S. (2009). Navigating the blogosphere: towards a genre-based typology of weblogs. *First Monday*, *14*(5).

Lund, K., Dyke, G., & Girardot, J.-J. (2009). Multimodal reformulation during shared synchronous note-taking and its potential pedagogical consequences for teachers and students. In *Actes d'EPAL 2009*. Grenoble, France.

Lund, K. (2007). The importance of gaze and gesture in interactive multimodal explanation. *International Journal of Language Resources and Evaluation*, *41*(3–4), 289–303.$

Lund, K., Molinari, G., Séjourné, A., & Baker, M. (2007). How do argumentation diagrams compare when student pairs use them as a means for debate or as a tool for representing debate? *International Journal of Computer-Supported Collaborative Learning*, *2*(2), 273–295.

Lund, K., Prudhomme, G., & Cassier, J.-L. (2007). Using analysis of computer-mediated synchronous interactions to understand co-designers' activities and reasoning. In *Proceedings of ICED'07*. Paris, France.

Martínez, A., Dimitriadis, Y., Rubia, B., Gomez, E., & de la Fuente, P. (2003). Combining qualitative evaluation and social network analysis for the study of classroom social interactions. *Computers and Education*, *41*(4), 353–368.

Martínez, A., de la Fuente, P., & Dimitriadis, Y. (2003). An XML-based representation of collaborative interactions. In *Proceedings of CSCL 2003* (pp. 379–384). Bergen, Norway.

Martínez, A. (2005). Library of interaction analysis methods. Deliverable of the ICALTS JEIRP.

Masseglia, F., Teisseire, M., & Poncelet, P. (2005). Sequential pattern mining: A survey on issues and approaches. *Encyclopedia of Data Warehousing and Mining*, 1028–1032.

May, M., George, S., & Prévôt, P. (2008). A closer look at tracking human and computer interactions in web-based communications. *International Journal of Interactive Technology and Smart Education*, *5*(3), 170–188.

McLaren, B. M., Scheuer, O., DeLaat, M., Hever, R., DeGroot, R., & Rosé, C. P. (2007). Using machine learning techniques to analyze and support mediation of student e-discussions. In *Artificial Intelligence in Education: Building Technology Rich Learning Contexts that Work* (pp. 331–338). Ios Pr Inc.

Mille, A. (2006). From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, *30*(2), 223–232.

Minsky, M. (1974). A framework for representing knowledge.

Mondada, L. (2006). La pertinenza del dettaglio : registrazione e trascrizione di dati video per la linguistica interazionale. In Y. Bürki & E. D. Stefani (Eds.), *Trascrivere la lingua. Dalla filologia all'analisi conversazionale* (pp. 313-344). Bern: Peter Lang.

Morrisson, A., Tennent, P., & Chalmers, M. (2006). Coordinated Visualisation of Video and System Log Data. In *proceedings of the 4th International Conference on Coordinated and Multiple Views in Exploratory Visualisation.* (pp. 91–102). London.

Mostow, J., & Aist, G. (2001). Evaluating tutors that listen: an overview of project LISTEN. In K. Forbus & P. Feltovitch (Eds.), *Smart machines in education* (pp. 169–234). Menlo Park, CA: MIT/AAAI Press.

Mostow, J., & Beck, J. (2006). Some useful tactics to modify, map, and mine data from intelligent tutors. *Natural Language Engineering (Special Issue on Educational Applications)*, *12*(3), 195–208.

Mostow, J., Beck, J., Cuneo, A., Gouvea, E., & Heiner, C. (2005). A generic tool to browse tutor-student interactions: Time will tell. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005)* (pp. 884–886).

Mühlenbrock, M. (2004). A computational model to differentiate between action and interaction in shared workspaces. In *The 2nd international workshop on designing computational models of collaborative learning interaction* (pp. 37–41). Maceió, Brasil.

Nüssli, M.-A., Jermann, P., Sangin, M., & Dillenbourg, P. (2009). Collaboration and abstract representations: towards predictive models based on raw speech and eye-tracking data. In *Computer Supported Collaborative Learning Practices: CSCL2009 Conference Proceedings*. Rhodes, Greece.

Overdijk, M., & van Diggelen, W. (2008). Appropriation of a shared workspace: organizing principles and their application. *ijCSCL*, *3*(2).

Piaget, J. (1976). *The grasp of consciousness: action and concept in the young child*. Cambridge, MA: Harvard University Press.

Prudhomme, G., Pourroy, & Lund, K. (2007). An empirical study of engineering knowledge dynamics in a design situation. *Journal of Design Research*, *3*, 333–358.

Psotka, J., Massey, L. D., & Mutter, S. A. (Eds.). (1988). *Intelligent Tutoring Systems: Lessons Learned*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Python Software Foundation (2000). Retrieved from http://docs.python.org/glossary.html#term-duck-typing.

Rabardel, P. (1995). *Les hommes et les technologies. Approche cognitive des instruments contemporains*. Paris: Armand Colin.

Reffay, C., & Chanier, T. (2003). How social network analysis can help to measure cohesion in collaborative distance learning. In *Designing for change in networked learning environments. Proceedings of the International Conference on Computer Support for Collaborative Learning* (pp. 343–352).

Reffay, C., Chanier, T., Noras, M., & Betbeder, M.-L. (2008). Contribution à la structuration de corpus d'apprentissage pour un meilleur partage en recherche. *STICEF*, *15*.

Reidsma, D., Jovanovic, N., & Hofs, D. (2005). Designing annotation tools based on properties of annotation problems. In *Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research, 30 August*. Technology.

Reimann, P. (2009). Time is precious: Variable- and event-centred approaches to process analysis in CSCL research. *ijcscl*, *4*(3), 239–257.

Reimann, P., Frerejean, & Thompson. (2009). Using Process Mining to Identify Models of Group Decision Making in Chat Data. In *Computer Supported Collaborative Learning Practices: CSCL2009 Conference Proceedings*. Rhodes, Greece.

Romero, C., Gutiérrez, S., Freire, M., & Ventura, S. (2008). Mining and visualizing visited trails in Web-based educational systems. In *Educational Data Mining 2008: 1st International Conference on Educational Data Mining, Proceedings* (pp. 182–186). Montréal, Canada.

Ronteltap, F., Goodyear, P., Bartoluzzi, S., & Limited, A. P. (2004). A Pattern Language as an Instrument in Designing for Productive Learning Conversations. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 4271–4276).

Rosé, C., Wang, Y.-C., Cui, Y., Arguello, J., Stegmann, K., Weinburger, A., et al. (2008). Analyzing collaborative learning processes automatically: Exploiting the advances of computational linguistics in CSCL. *ijCSCL*, *3*(3).

Rummel, N., Meier, A., Spada, H., Karimanis, G. & Avouris, N. (in press). Analyzing collaborative interactions across domains and settings: An adaptable rating scheme. To appear in S. Puntambekar, C. Hmelo-Silver & G. Erkens (Eds.), *Analyzing interactions in CSCL: Methodology, approaches and issues*. Berlin: Springer

Russell, Stefik, Pirolli, & Card. (1993). The cost structure of sensemaking. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems* (pp. 269–276). New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/169059.169209.

Sacks, H., Schegloff, E. A., & Jefferson, G. (1974). A simplest systematics for the organisation of turn-taking for conversation. *Language*, *50*, 696–735.

Sanderson P., & Fisher C. (1994). Exploratory sequential data analysis: foundations. *Hum.-Comput. Interact.*, *9*(4), 251–317.

Sanderson, P., Scott, J., Johnston, T., Mainzer, J., Watanabe, L., & James, J. (1994). MacSHAPA and the enterprise of exploratory sequential data analysis (ESDA). *International Journal of Human-Computer Studies*, *41*(5), 633–681.

Sanderson (1994). *MacSHAPA version 1.0.2 Manual.* Available online: http://www.itee.uq.edu.au/~macshapa/

Savoie-Zajc, L. (2000). L'analyse des données qualitatives : Pratiques traditionnelles et assistées par le logiciel NUD*IST. *Recherches Qualitatives*, *21*, 99–123.

Scardamalia, M., & Bereiter, C. (1991). Higher levels of agency for children in knowledge building: A challenge for the design of new knowledge media. *The Journal of the Learning Sciences*, *1*(1), 37–68.

Scardamalia, M., & Bereiter, C. (2003). Knowledge building environments: extending the limits of the possible in education and knowledge work. In A. DiStefano, K. E. Rudestam, & R. Silverman (Eds.), *Encyclopedia of distributed learning*. Thousand Oaks, CA: Sage Publications.

Schank, R. C. (1982). *Dynamic memory: a theory of reminding and learning in computers and people*. Cambridge University Press.

Settouti, L. S., Prié, Y., Champin, P. A., Marty, J.-C., & Mille, A. (2009). A Trace-Based Systems Framework: Models, Languages and Semantics. *Published on HAL-CCSD.*

Settouti, L. S., Prié, Y., Mille, A., & Marty, J.-C. (2006). Système à base de traces pour l'apprentissage humain. In *Colloque TICE 2006*.

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Springer-Verlag.

Stahl, G. (2009). *Studying virtual math teams*. Springer.

Strijbos, J., Martens, R. L., Prins, F. J., & Jochems, W. M. (2006). Content Analysis: what are

they talking about? *Computers & Education*, *46*, 29–48.

Strijbos, J. W., & Fischer, F. (2007). Methodological challenges for collaborative learning research. *Learning and Instruction*, *17*(4), 389–393.

Suthers, D. (2006). Technology affordances for intersubjective meaning making: A research agenda for CSCL. *ijCSCL*, *1*(3), 315–337.

Suthers, D., Dwyer, N., Medina, R., & Vatrapu, R. (2007). A Framework for Analyzing Interactional Processes in Online Learning. In *Annual Meeting of the American Educational Research Assiciation (AERA)*. Chicago.

Suthers, D., & Medina, R. (2008). Tracing Interaction in Distributed Collaborative Learning. In *Annual Meeting of the American Educational Research Assiciation (AERA)*. New York.

TEI (2007). Retrieved from http://www.tei-c.org.

Teplovs, C., Green, A., & Scardamalia, M. (2008). The ZooLib Tuplebase: An Open-Source, Scalable Database Architecture for Learning Sciences Research. In *Proceedings of the International Conference of the Learning Sciences 2008*. Utrecht, NL.

Teplovs, C. (2008). The Knowledge Space Visualiser: a tool for visualising online discourse. In *Common Framework for CSCL Interaction Analysis Workshop at ICLS2008*. Utrecht, The Netherlands.

Tesch, R. (1990). *Qualitative Research: Analysis Types and Software Tools*. The Falmer Press.

Teutsch, P., Bangou, F., & Dejean-Thircuir, C. (2008). Faciliter l'accès aux échanges en ligne et leur analyse, le cas de ViCoDiLi. *Revue Sticef*, *15*.

Thomas, J. J., & Cook, K. A. (Eds.). (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE.

de Vries, E., Lund, K., & Baker, M. J. (2002). Computer-mediated epistemic dialogue: Explanation and argumentation as vehicles for understanding scientific notions. *The Journal of the Learning Sciences*, *11*(1), 63–103.

W3C (2007). XQuery 1.0: An XML Query Language. Retrieved from http://www.w3.org/TR/xquery/

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge Univ Pr.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge*. Los Altos, CA: Morgan Kaufmann.

de Wever, B., Schellens, T., Valcke, M., & Van Keer, H. (2006). Content analysis schemes to

analyse transcripts of online asynchronous discussion groups: a review. *Computers & Education*.

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., & Sloetjes, H. (2006). Elan: a professional framework for multimodality research. In *Proceedings of Language Resources and Evaluation Conference (LREC)*.

Zemel, A. and Çakir, M. (2009). Reading's work in VMT. *Studying virtual math teams*.

Publications

International conferences with peer review

Dyke, G., Lund, K., Girardot, J.-J. (2009). Tatiana: an environment to support the CSCL analysis process. *CSCL 2009*, Rhodes, Greece.

Lund, K., van Diggelen, W., Dyke, G., Overdijk, M., Girardot, J.J., & Corbel, A (2008). A researcher perspective on the analysis and presentation of interaction log files from CSCL situations within the LEAD project. *ICLS 2008*, Utrecht, The Netherlands.

National conferences with peer review

Dyke, G., Beigbeder, M., Lund, K., Girardot, J.-J. (2009). Les Traces d'interactions humaines : un nouveau domaine d'application pour la RI. *CORIA 2009*, Toulon, France.

Lund, K., Dyke, G. Girardot, J.-J. (2009). Multimodal reformulation during shared synchronous note-taking and its potential pedagogical consequences for teachers and students. *EPAL 2009*, Grenoble, France.

Dyke, G., Lund, K. (2007). Implications d'un modèle de coopération pour la conception d'outils collaboratifs. *EPAL 2007*, 6-8 Juin, Grenoble, France.

Conferences without proceedings

Dyke, G., Lund, K., Girardot, J.-J. (2009). Un outil flexible pour l'analyse de traces - Gestion, synchronisation, visualisation, analyse et partage de corpus d'interactions médiatisées par ordinateur avec Tatiana. *Atelier ICT 2009*, Caen, France.

Dyke, G., Girardot, J.-J., Lund, K., Corbel, A (2007). Analysing Face to Face Computer-mediated Interactions. *EARLI '07*, Budapest, Hungary.

Demonstrations and workshops

Dyke, G., Lund, K., Girardot, J.-J. (2009). La visualisation au service de l'analyse d'interactions. *Poster à EIAH 2009*, Le Mans, France.

Dyke, G., Lund, K., Girardot, J.-J. (2009). Using the analysis tool Tatiana to visualize and calculate interaction indicators. *Interaction Analysis Workshop, CSCL 2009*, Rhodes, Greece.

Dyke, G., Lund, K., Girardot, J.-J. (2008). Managing, synchronising, visualising, analysing and sharing multimodal computer-mediated human interaction data: introducing Tatiana. *A Common Framework for CSCL Interaction Analysis, workshop at ICLS 2008*, Utrecht, The Netherlands.

Journal publications currently under review

- Dyke, G. Lund, K., Girardot, J.-J. (submitted to Technique et Science Informatiques). « Tatiana : un environnement d'aide à l'analyse de traces d'interactions humaines. »

- Dyke, G. Lund, K., Girardot, J.-J. (submitted to STICEF). « L'étude des interaction humaines au travers des objets rejouabales. »

Appendix I – Tatiana Architecture

In Fig 1, we present a subset of the classes of Tatiana which is relevant to understanding the mechanisms of synchronisation, visualisation and enrichment.

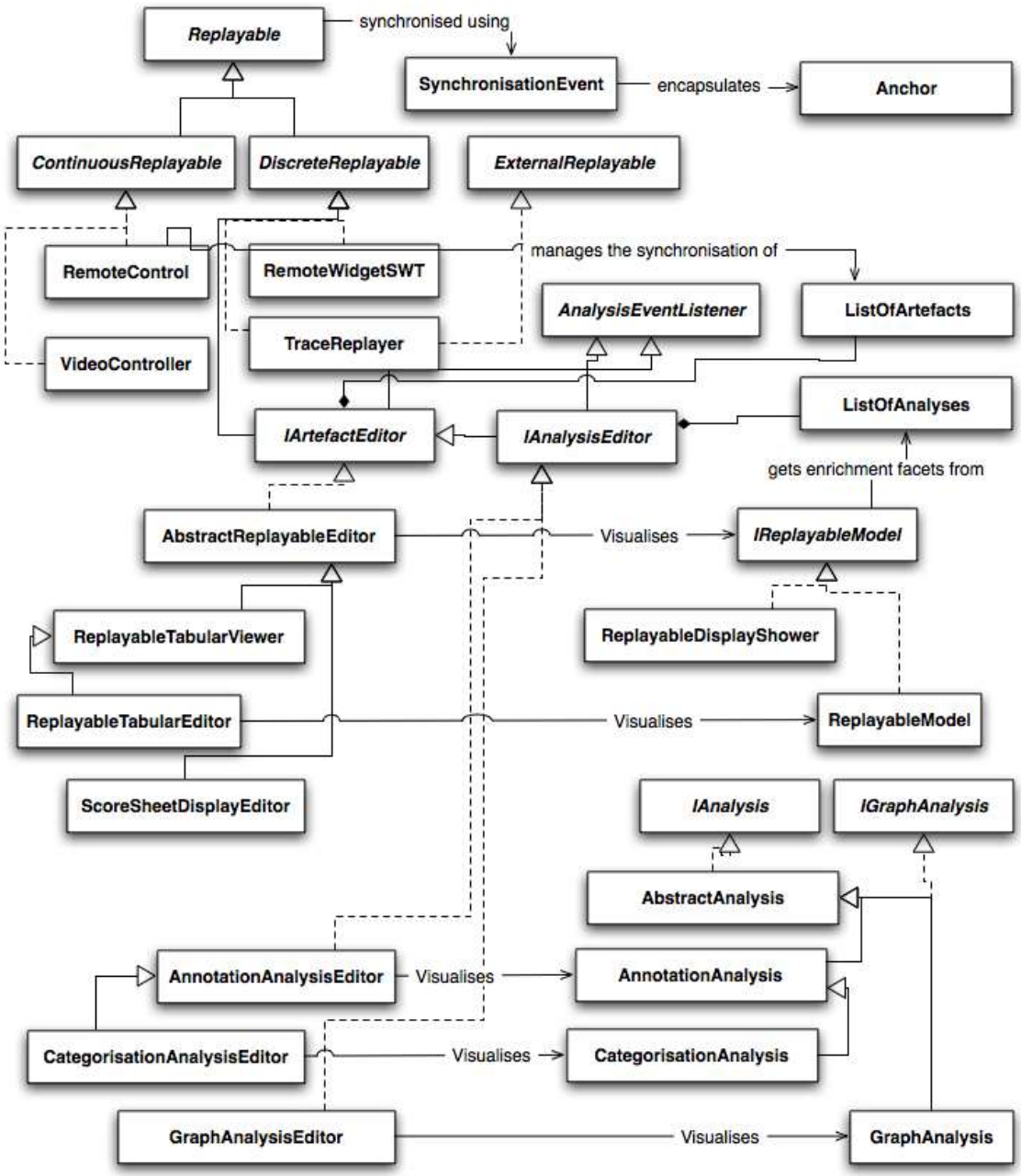


Fig 1 Tatiana Class Diagram

Synchronisation is applied across all implementers of the Replayable interface which can be broken up into five classes (and interfaces). The RemoteControl (a singleton class), which is the central management system for synchronisation and replay, and the VideoController, which provides the interface for the reading of multimedia files, both implement ContinuousReplayable, and accept "play" and "pause" commands. The RemoteWidgetSWT, which is the graphical user interface for managing the remote control, the TraceReplayer, which provides the interface for communicating with external replayers (and thus implements ExternalReplayable) and the various implementers of IArtefactEditor (which are visualisations of replayables and enrichments). In synchronisation, the duality of simultaneity and identity is managed by encapsulating synchronisation information in a SynchronisationEvent, which in turn makes use of the Anchor class (and also describes a time interval).

Replayables implement the IReplayableModel interface and currently have two kinds of implementations: replayables which are directly editable (ReplayableModel) and others are only editable by editing the transformation that creates them (ReplayableDisplayShower – named for historical reasons). A list of open analyses is maintained and from that list, replayables obtain their enrichment facets (also communicating using instances of Anchor – a link which is not shown in Fig 1).

Replayable Visualisations implement the IArtefactEditor interface and should also implement Eclipse's IEditor interface (not shown in Fig 1). A base implementation, AbstractReplayableEditor, is provided. Replayable visualisations should be able to visualise implementations of the IReplayableModel interface. An exception are replayable visualisations which are intended to be able to directly modify a replayable (such as ReplayableTabularEditor, the editable tabular visualisation) which may assume they are visualising an instance of ReplayableModel.

Transformation is not shown in Fig 1 and is handled by ReplayableTabularViewer. This editor provides a user interface for selecting filters and parameters for these filters. As in all Eclipse editors, the model to be viewed is communicated to the editor via an IEditorInput. In Tatiana, this is an IReplayableInput, which abstracts away the difference between the two kinds of replayables (one of which is opened directly, while the other describes a transformation which must be run – or whose result can be obtained from the cache).

Enrichment editors implement the IAnalysisArtefact interface and should be designed to model their own kind of enrichment, which must implement the IAnalysis. **Enrichments**

which provide binary relationships should also implement IGraphAnalysis. A base implementation of IAnalysis (AbstractAnalysis) is provided. Unlike replayable visualisations which mostly assume that they are visualising an IReplayableModel, enrichment visualisations are intrinsically associated with their own analysis model.

Appendix II – Tatiana info files and the Replayable API

Introduction

The Tatiana pivot format (known as "info format") is used to describe event sequences (also known as "replayables") and other data in Tatiana. It constitutes an XML common generic representation which is used for many purposes:

* Corpus description;
* Replayables description, constituted by sequences of events;
* Filter descriptions (see WritingFilters);
* Analyses description, etc.

Overview

Basically, a display file contains a sequence of items, and each item contains a set of infos. As the names suggest, items of a display file represent an ordered sequence, while info elements inside an item are unordered. Conceptually, each item is an independent object, while each info element describes a facet of this object. Each info element has a name and a value. Values are typed, representing either very classical data-type (boolean, integer, text) or Tatiana specific data-types, such as time or anchor.

Syntactically, a display file is defined by the following DTD (note that the entirety of this DTD has not yet been implemented in Tatiana):

```
<!ELEMENT display (item)*>
<!ELEMENT item (info)*>
<!ENTITY % infocontent
"#PCDATA|time|anchor|item|text|icon|img|
    color|bgcolor|integer|number|boolean|seq|set|html|element|
display" >
```

```
<!ELEMENT info        (%infocontent;)*>
<!ATTLIST info
        name CDATA #REQUIRED >
<!ELEMENT time                  (date, (date|duration)?)>
     <!ELEMENT date                  (#PCDATA)>
     <!ELEMENT duration              (#PCDATA)>
<!ELEMENT anchor              (file|event|path|doc)+>
     <!ELEMENT doc (#PCDATA)>
     <!ELEMENT event (#PCDATA)>
<!ATTLIST event
     elt-name NMTOKEN #REQUIRED >
     <!ELEMENT file (#PCDATA)>
     <!ELEMENT path (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ATTLIST text
        color CDATA #IMPLIED
        bgcolor CDATA #IMPLIED >
<!ELEMENT img  EMPTY>
<!ATTLIST img  src CDATA #REQUIRED >
<!ELEMENT icon EMPTY>
<!ATTLIST icon src CDATA #REQUIRED >
<!-- Basic data types -->
<!ELEMENT boolean (#PCDATA)>
    <!-- content should be true,false, 0 or 1 -->
<!ELEMENT integer (#PCDATA)>
    <!-- content is the decimal representation of an integer -
->
```

Replayables

A replayable is stored as an info file that contains a sequence of events (or Tatiana Display File). Each event is defined by an item, that contains at least an info element with name "time", and which value is of type time. If this replayable has been created from a trace file (whatever tool produced the trace : a CSCL/CSCW tool such as DREW or CoFFEE or a similar tool, a log file, etc.), the item also contains an info element, whose name is "src-anchor", and whose value is of type “anchor”. These two particular facets are used by Tatiana when managing replayables. An item can also contain any additional number of info elements, which describe facets of the event, like the tool that created the event, the user involved, the nature of the event, text produced by the user, or any other significative information.

Within Tatiana, replayables can be accessed using the API defined in IReplayableModel:

> Model for representing and accessing replayables. Note that we do not specify the class for representing events. The information encapsulated in an event should be accessed through the IReplayableModel instance. All facet values are accessed through the position of that facet in a list of facets.

Method Summary	
java.util.List<java.lang.Object>	**anchoredEvents**(fr.emse.tatiana.replayable.SynchronisationEvent event) get the events within the set of anchors encapsulated by the synchronisation event (identity)
boolean	**canModifyColumn**(int col) Is the given facet editable
java.util.List<java.lang.Object>	**currentEvents**(fr.emse.tatiana.replayable.SynchronisationEvent event) get the events within the interval encapsulated by the synchronisation event (simultaneity)
void	**doSave**() Save this replayable
void	**doSave**(java.lang.String value) Save this replayable under a new name
java.lang.Object[]	**getAnalysedEvents**(fr.emse.tatiana.replayable.analysis.AnalysisEvent evt) Find the events which have been analysed under a given analysis event (presumably to update the UI)
java.lang.O	**getAnalysedEvents**(fr.emse.tatiana.replayable.analysis.IAna

bject[]	lysis a) Find the events which have been analysed under analysis a.
java.util.List<fr.emse.tatiana.replayable.analysis.IAnalysis>	**getAnalyses**() Get the analyses applied to the replayable
fr.emse.tatiana.replayable.Anchor	**getAnchor**(java.lang.Object o) Get the src-anchor for a given event
org.eclipse.swt.graphics.Color	**getBackground**(java.lang.Object element, int columnIndex) Get the background color for a facet of an event
long	**getBeginTime**(java.lang.Object firstElement) Get the time at which the event begins
org.eclipse.jface.viewers.CellEditor	**getCellEditor**(int col, org.eclipse.swt.widgets.Composite c) Return the GUI widget for editing a given facet
java.lang.Object	**getEvent**(int index) Get event at row index
int	**getEventCount**() Number of events in the replayable
java.util.List<java.lang.Long>	**getEventTimestamps**() Get all the timestamps of the events in the replayable (including begin and end of intervals)
java.lang.Object	**getFacet**(java.lang.Object elem, int col) Get a facet of an event
int	**getFacetCol**(java.lang.String name) Gets the number of the column for the given facet name Don't call this method with the facets "time" or "src-anchor" (as they are not really facets) This is the inverse of getFacetName (the column matching that name)
int	**getFacetCount**() Get the total number of facets in this replayable.
int	**getFacetIdFor**(fr.emse.tatiana.replayable.analysis.IAnalysis a) Gets the number of the column for a given analysis.
java.lang.String	**getFacetName**(int col) Get the name of the facet represented under the given column
java.lang.String	**getFacetText**(java.lang.Object element, int columnIndex) Text value for a facet of an event
long	**getFirstTime**() Get the smallest timestamp in the replayable (not necessarily the first event)

org.eclipse.swt.graphics.Color	**getForeground**(java.lang.Object element, int columnIndex) Get the foreground color for a facet of an event
long	**getLastTime**() Get the largest timestamp in the replayable (not necessarily the first event)
long	**getLastTime**(java.lang.Object object) Get the time at which the event ends
java.lang.String	**getName**() get the name of the replayable
boolean	**hasReplayableFile**() Has the replayable been saved to disk at least once?
boolean	**isAnalysisFacet**(int i) Is a given column (representing a facet) an analysis column?
java.lang.Object	**lastEventBefore**(long time) find the event whose timestamp is neares the time
void	**setFacetValue**(java.lang.Object event, int col, java.lang.Object value) Assign a facet of an event

Appendix III – Tatiana enrichment API

Enrichment plugins should provide both a model (an implementation of IAnalysis) and an Eclipse editor for viewing and editing this model (an implementation of IAnalysisEditor). Currently, analyses can only provide a single additional facet and a special form of analysis (IGraphAnalysis) must be used to provide binary relationships between events.

`public interface` **`IAnalysis`**

Method Summary	
`void`	**`addAnalysisEventListener`**`(AnalysisEventListener listener)` Register as a listener for changes to this analysis
`void`	**`doSave`**`()` Save the analysis
`void`	**`doSave`**`(java.lang.String analysisname)` Save the analysis with a given name
`java.util.Set<fr.emse.tatiana.replayable.Anchor>`	**`getAnchors`**`()` Get the set of events (defined by their anchors) which are enriched in this analysis
`org.eclipse.swt.graphics.Color`	**`getBackground`**`(fr.emse.tatiana.replayable.Anchor anchor)` Get the background color for the facet which enriches a given event
`org.eclipse.jface.viewers.CellEditor`	**`getChooser`**`(org.eclipse.swt.widgets.Composite parent)` Get the GUI widget to choose the enrichment value
`java.lang.String`	**`getName`**`()` Get the name of the analysis (filename and name of the facet)
`java.lang.String`	**`getTextValueForAnchor`**`(fr.emse.tatiana.replayable.Anchor anchor)` Get the text to display for the facet for a given anchor
`java.lang.Object`	**`getValueForAnchor`**`(fr.emse.tatiana.replayable.Anchor anchor)` Get the facet value for a given anchor
`int`	**`getWidth`**`(int col)` Shouldn't really be here, but provides a hint at how wide the column

	needs to be
boolean	**hasAnalysisFile**() Has the analysis been saved to file at least once
boolean	**hasValueForAnchor**(fr.emse.tatiana.replayable.Anchor anchor) Test whether an event is enriched in the analysis (the event is identified by its Anchor)
boolean	**isSaved**() Has the analysis been modified recently
void	**removeAnalysisEventListener**(AnalysisEventListener listener) Unregister as a listener to changes to this analysis
void	**setValue**(fr.emse.tatiana.replayable.Anchor anchor, java.lang.Object value) Enrich an event (defined by its anchor) with a given value

```
public interface IGraphAnalysis
extends fr.emse.tatiana.replayable.analysis.IAnalysis
```

This interface represents analyses which provide binary relationships between events. These relationships are represented by Link objects which encapsulate the anchors of the source and target as well as the type of relationship.

Method Summary	
void	**createLink**(fr.emse.tatiana.replayable.Anchor source, fr.emse.tatiana.replayable.Anchor target, int type, java.lang.String desc) Create a new relationship
java.util.List<fr.emse.tatiana.replayable.analysis.Link>	**getLinks**() Get all the relationships in this analysis.
java.util.List<fr.emse.tatiana.replayable.analysis.Link>	**getSourceConnections**(fr.emse.tatiana.replayable.Anchor anchor) Get all the relationships for which an event is the source
java.util.List<fr.emse.tatiana.replayable.analysis.Link>	**getTargetConnections**(fr.emse.tatiana.replayable.Anchor anchor) Get all the relationships for which an event is the target
void	**removeLink**(fr.emse.tatiana.replayable.analysis.Link link) Delete a relationship
void	**updateLink**(fr.emse.tatiana.replayable.analysis.Link link) Notify that a relationship has been modified

Methods inherited from interface fr.emse.tatiana.replayable.analysis.IAnalysis
addAnalysisEventListener, doSave, doSave, getAnchors, getBackground, getChooser, getName, getTextValueForAnchor, getValueForAnchor, getWidth, hasAnalysisFile, hasValueForAnchor, isSaved, removeAnalysisEventListener, setValue

public interface **IAnalysisEditor**
extends IArtefactEditor

Method Summary	
void	**analysisEventOccurred**(fr.emse.tatiana.replayable.analysis.AnalysisEvent evt) See interface AnalysisEventListener
IAnalysis	**getAnalysis**() Get the analysis model represented by this editor

Methods inherited from interface fr.emse.tatiana.replayable.IArtefactEditor
addAnalysis, removeAnalysis, setRemote

Methods inherited from interface fr.emse.tatiana.replayable.Replayable
end, getEvents, goTo, mark

Classes that wish to be notified of changes to analyses can implement IAnalysisListener. Implementations of IAnalysis should maintain a list of listeners and send AnalysisEvent instances to describe modifications.

skip-navbar_top
public interface **AnalysisEventListener**
extends java.util.EventListener

Method Summary	
void	**analysisEventOccurred**(fr.emse.tatiana.replayable.analysis.AnalysisEvent evt)

Appendix IV – Tatiana synchronisation API

API for plugins

In Tatiana, objects which benefit from the synchronisation mechanism must implement the Replayable interface. In particular, plugins should implement the IArtefactEditor interface, which is an extension of DiscreteReplayable, which is an alias for Replayable. IArtefactEditor also extends AnalysisEventListener, allowing Editors to be notified of analysis changes.

```
public interface IArtefactEditor
extends AnalysisEventListener, DiscreteReplayable
```

Represents any kind of visualisation which can be synchronised in Tatiana, including Replayables and Enrichments. Such visualisations may also want to include notification about analysis events.

Method Summary	
void	**addAnalysis**(IAnalysis a) Add an analysis to the list of analyses managed by the editor
void	**removeAnalysis**(IAnalysis a) Remove an analysis from the list of analyses managed by the editor
void	**setRemote**(fr.emse.tatiana.remotecontrol.RemoteControl remote) Provide the editor with the RemoteControl object which it should use for synchronisation

Methods inherited from interface fr.emse.tatiana.replayable.analysis.AnalysisEventListener
analysisEventOccurred

Methods inherited from interface fr.emse.tatiana.replayable.Replayable
end, getEvents, goTo, mark

skip-navbar_top

```
public interface Replayable
```

Method Summary	
void	**end**() Notify the replayable that it no longer receives synchronisation events.
java.util.List<java.lang.Long>	**getEvents**() Request the list of timestamps for which the replayable wants to be notified.
void	**goTo**(fr.emse.tatiana.replayable.SynchronisationEvent event) Position the replayer at this replayable unit.
void	**mark**(long time) Not currently used

API for external replayers

What is a replayer?

A replayer is a tool that takes a trace file produced by a piece of software and "replays" it by reproducing on screen what the user saw (or some approximation or enhancement thereof).

Why create a replayer that is pilotable by Tatiana?

Tatiana is a tool whose aim is to assist researchers in analysing data produced by a variety of recording mechanisms, including that of tracing. Tatiana aims to be very flexible and being able to read any kind of trace file (provided it is in XML format). It leaves understanding the contents of this file to the researcher. Often, even this is not enough, hence the use of a replayer. A replayer provides the researcher with extra context.

This context is even more useful if it is synchronized with Tatiana's presentation of the data. In order to do this, Tatiana has a communication protocol for talking to external replayers. Finally, replayer writers who choose to implement this protocol can take advantage of our lightweight remote control in order to quickly implement a simple standalone replayer.

What messages does a replayer need to understand to be pilotable by Tatiana?

Replayer implementations must accept the following messages:

- `openFile`(filename) -- Open the specified trace file.

- `goto`(timestamp) -- Position the replayer at the given timestamp.

- `mark`(timestamp) -- Notify the replayer that this timestamp may be frequently returned to.

- `end`() -- Tell the replayer to exit.

- `getEvents`() -- Return a list of timestamps present in the currently opened trace file.

Replayers may also accept the following set of commands:

- `play`() -- Start playing the trace at the current speed

- `pause`() -- Stop playing the trace

- `setSpeed`(speed) -- Set the speed of replay

If they do not, Tatiana will handle the flow of time, sending goto to the replayer at the appropriate time. In the second case, replayers handle time on their own.

We call the first case a discrete replayer. The second is a continuous replayer.

How do I implement a replayer?

Tatiana communicates with replayers via XML-RPC. Tatiana must be able to launch the replayer, specifying a port. The replayer will create an xml-rpc server on http://localhost:<port>/tatiana. This allows multiple instances to be launched simultaneously.

We provide classes for easy implementation of replayers in Java. The requirements for each message are as follows:

`openFile`

parameters

-filename: string - path to file to be opened

returns

-boolean - indicates success or failure

result

The replayer parses the specified file. The replayer is now ready to respond to a getEvents message, or to a play/pause/setSpeed message.

notes

Only one file may be opened at a time. openFile may be taken as an indication to finish replaying the previous file and to close related windows.

`goTo`

parameters

- timestamp: string - string equivalent of a UNIX timestamp

returns

- boolean - indicates success or failure

result

The replayer executes all events in the trace file whose timestamps are smaller or equal to the provided timestamp. In other words, the replayer positions itself at the given timestamp.

notes

Accuracy to at least 1/10s is desirable. Xml-rpc's dateTime.iso8601 does not provide sufficient precision. Xml-rpc's int is 32-bit - while this is sufficient to represent UNIX timestamps, we prefer to err on the side of safety as implementations of xml-rpc may not always be 32-bit.***THIS MAY CHANGE IN THE FUTURE***

There is no guarantee that this timestamp corresponds to an event or that it is within "range" for the current trace file. Continuous replayers should "remember" that they are in "play"/"pause" mode, even when they are out of range. Note that goTo's to the "past" are a likely occurrence.

`mark`

parameters

- timestamp: string - string equivalent of a UNIX timestamp

returns

- boolean - indicates success or failure

result

The replayer prepares itself for the possibility that it may soon receive a goTo for the same timestamp.

notes

This is currently provided for optimisation only. Replayers may ignore this command - if they do so, they should return 0 (false).

`end`

returns

- boolean - indicates success or failure

result

The replayer exits.

notes

Replayers are under no obligation to return from this procedure before exiting.

`getEvents`

returns

- array[string] - a list of the timestamps of events in the file. Each element in the list is a string equivalent of a UNIX timestamp.

notes

This may change to array[int] in the future. The timestamps need not be ordered. Tatiana will manage the flow of time and send goTo's to the replayer when it reaches the timestamps returned by this message.

`play`

returns

- boolean - indicates success or failure

result

The replayer puts itself in "play" mode.

`pause`

returns

- boolean - indicates success or failure

result

The replayer puts itself in "pause" mode.

notes

Replayers are assumed to begin life in pause mode.

`setSpeed`

parameters

- speed: int - speed at which time should "flow"

returns

- boolean - indicates success or failure

result

The replayer sets its playing speed. This speed will be a positive non-zero integer. Normal speed is 1, double speed is 2, etc.

How do I implement a replayer in java using TatianaRemoteService.jar?

TatianaRemoteService uses the xml-rpc client-server by Apache.

Replayers must extend the abstract class DiscreteReplayerService (or ContinuousReplayerService) found in TatianaRemoteService.jar

Creating an instance of this class will automatically set up an xml-rpc server on the specified port.

Note that mark and goTo use timestamps of type Long. Both replayer services include the conversion.

Appendix V – Authoring scripts and filters

About scripts and filters

In Tatiana, replayables are created through the use of *filters*. Filters are objects which combine *scripts* into a workflow. Scripts are small programs written to perform a specific operation, such as transform a file in the corpus into data Tatiana can understand, exclude certain kinds of events from a replayable, find certain kinds of events in a replayable, combine multiple replayables, etc. As we only expect to write these scripts, we are currently developing a graphical editor for filters which will allow researchers to customize the execution of these scripts. Such a filter might combine a new script for data import from the interaction log data produced by a new kind of tool with an existing script which only shows the actions of a particular subset of students.

In order to execute even one script only, it is currently necessary to create a filter (which currently must be created by hand) which calls this script.

Adding new filters and scripts

Tatiana filters must be placed in the folder `TatianaWorkspace/Tatiana/filters`. The filename must end in `.xml`. They are written in an internal XML format as described in this document.

Tatiana scripts must be placed in the folder `TatianaWorkspace/Tatiana/scripts`. The filename must end in `.xq`. They are written in XQuery.

Scripts

Reminder on the Tatiana Display Format

A file in the Tatiana Display Format (see DisplayFormat) is an XML document whose root tag is `<display>` (named thus for historic reasons). It is composed of `<item>`s (representing events) which are in turn composed of `<info>`s (the properties of these events). An `<info>` is a named and typed property. The default type is a String and conversion between types is done on an unchecked pragmatic basis. An information on the tool producing and event might be written:

```
<info name="tool">drew-chat</info>
```

The names of properties are defined in a completely ad-hoc manner. You are free to give the names you want to the properties of the events that you choose to trace. Ideally you would include information identifying objects, participants, generating tools, type of the event, etc.

Two compulsory properties are the "source-anchor" which must be of type "anchor" and "time" which must be of type "time". This means that everything that is contained in a `<info name="source-anchor">` must in turn be contained within a single `<anchor>` tag. For a given event (`<item>`) in a corpus, its anchor *must* be unique (we suggest using a tuple describing the name of the document and the path within that document). Everything that is contained within a `<info name="time">` must in turn be included within a single `<time>` tag.

```
<info name="time">
    <time>
      <date>1166547729747</date>
      <duration>1637</duration>
    </time>
</info>
```

The date is a unix timestamp. The duration is in milliseconds.

Introduction to XQuery

In order to write scripts for Tatiana, one must first learn XQuery. We do not have any recommendations beyond reading the W3C specification. XQuery is a functional programming language which is an extension of XPath allowing iteration through sequences

and generation of XML output. This presentation of XQuery is in no way comprehensive but introduces some of the things that can be done with XQuery. In particular, we do not cover the XQuery function API which is very rich and powerful.

XQuery has only constructors (XML constructors in particular) and expressions. The basic expression is an XPath expression (we will not describe XPath here). The expression that is particular to XQuery is the FLWOR expression, which generates a new sequence of values by iterating through another sequence of values.

F **`for`** `$variable` **`in`** `expression` - iterate through the sequence generated by the expression

L **`let`** `$variable` **`:=`** `expression` - assign a expression to the variable (note that XQuery variables are not mutable). You may assign any number of $variable before or after the "for"-part. Variables assigned before will be global to the FLWOR expression. Variables assigned after will be specific to each value in the iteration sequence.

W **`where`** `expression` - only evaluate where expression is true (optional)

O **`order by`** `expression` - the return order is lexical or numerical ordering of the expression. If an order is not specified, the order is that of the original sequence (optional).

R **`return`** `expression` - value to be yielded at each iteration. The result of the FLWOR expression is a sequence of the values returned by the "return" part of the expression.

XQuery has the datatypes of XPath: Some XML types, some atomic types, and sequence types. Sequences may only be composed of atomic types. A sequence of sequences will be "flattened out". A particular use of this is that for an iteration of a FLWOR expression to yield two elements, they should yield those elements in a sequence. And in order for an iteration *not* to yield an element, it should return the empty sequence. In other words, the expression `(1, (2, 3), 4, (), 5)` is equal to the expression `(1, 2, 3, 4, 5)`.

A first example of a transformation

Here is the script which is used to merge any number of display files together, preserving ordering of events in time.

```
<display>{
  for $item in (for $d in $arguments return
doc($d)/display/item)
  order by $item/info[@name="time"]/time/date[1]
  return $item
```

```
}</display>
```

Let us examine this script expression by expression.

```
doc($d)/display/item
```

Opens the document with the filepath contained in $d. Retrieve the sequence of <item>s (events) contained within the <display> tag (the root of the document).

```
for $d in $arguments return doc($d)/display/item
```

This FLWOR expression iterates through the filenames contained in the $arguments sequence (which is a variable particular to the Tatiana XQuery engine, which allows passing parameters to XQuery scripts) and for each file returns all the items within that file. Because sequences of sequences are flattened, this expression returns a single sequence containing all the items found in all the files.

```
  for $item in (for $d in $arguments return
doc($d)/display/item)
  order by $item/info[@name="time"]/time/date[1]
  return $item
```

This FLWOR expression iterates through all these items and orders them by accessing the date in the "time" info.

```
<display>{
  for $item in (for $d in $arguments return
doc($d)/display/item)
  order by $item/info[@name="time"]/time/date[1]
  return $item
}</display>
```

This expression is a constructor which constructs a <display> element whose children will be all the nodes in the sequence contained within the brackets. In general, the construction of an XML file will require that the main expression of an XQuery script be the constructor of the root XML element.

A second example of a transformation

Here is a script which numbers each of the events in a display file by adding a property named "num" to each of the events.

```
<display>{
let $fpath := $arguments[1]
let $items := doc($fpath)/display/item
for $item at $p in $items
return
  <item>{
    <info name="num">{ $p }</info>,
```

```
    (  let $infos := $item/info
       for $info in $infos
       return
         if ($info[@name="num"]) then
             ()
         else
             $info
    )
  }</item>
}</display>
```

As before, the root expression is a constructor of a `<display>` element. The children of this element are generated by the main FLWOR expression (two lets, one for, one return), which names the first argument passed to the script "$fpath", names the items in that document "$items", and iterates through those items. The expression yielded by each item is a new item whose contents we will now examine in greater detail.

```
  <item>{
    <info name="num">{ $p }</info>,
    (  let $infos := $item/info
       for $info in $infos
       return
         if ($info[@name="num"]) then
             ()
         else
             $info
    )
  }</item>
```

The returned `<item>` is a sequence of elements, the first of which being: (note that $p is assigned in a variant of the "for" statement)

```
    <info name="num">{ $p }</info>
```

The others are the elements returned by the FLWOR expression

```
    (  let $infos := $item/info
       for $info in $infos
       return
         if ($info[@name="num"]) then
             ()
         else
             $info
    )
```

which copies all of the properties of the event, *except* for the properties named "num" - this allows two numbered replayables to be merged and renumbered without having any duplicate "num" properties. Note that `$info[@name="num"]` evaluates as true if the sequence it returns is not empty.

Alternative ways of writing this expression (using increasingly fewer XQuery constructs and letting XPath do the work) might be

```
for $info in $item/info
where $info[@name!="num"]
return $info
for $info in $item/info[@name!="num"]
return $info
```

or even

```
$item/info[@name!="num"]
```

In other words, for each event in the original file, this script returns an event with all the original event's properties (except for the one called "num", if it exists) and containing an extra "num" property.

A third example of a transformation

This script takes a display file, a property name and a property value as parameters. It returns only those events where the property of that name has that value.

```
<display>{
let $infoname := $arguments[2]
let $infovalue := $arguments[3]
let $t := $arguments[1]
for $item in doc($t)/display/item
        where some $info in $item/info satisfies $info/@name =
$infoname and $info/text()=$infovalue
        return $item
}</display>
```

It is straight forward enough, and introduces another XQuery expression which is a nicer way of expressing

```
where $item/info[@name=$infoname][./text()=$infovalue]
```

A first example of an import script

This script is designed to convert a DREW trace file into a Tatiana display file. It only selects the chat events (we assume the general pattern of "import everything" to be easier to write as "import type A", "import type B", "import type C", "merge the three previous results"). The element we wish to import looks something like this in the DREW XML:

```
<event user="john" room="Hall">

<time><date>1193316307909</date><duration>10837</duration></ti
me>
```

```
  <chat><text lang="en">hello</text></chat>
</event>
```

The script selects only the <event>s which contain a <chat> element (DREW has other modules which produce other elements in the trace). It then extracts the user, time, room and content and creates an event with these properties. It also adds in the "type" and "src-anchor" properties.

```
import module namespace
    jj = "http://kumquat.emse.fr/utilitaires"
    at "jjutils.xq" ;
<display>{
let $t := $arguments[1]
let $d := doc($t)/trace/event
for $e at $p in $d
where $e[chat]
return
  <item>{
    <info name="type">drew-chat</info> ,
    <info name="time">{ $e/time }</info>,
    <info name="src-anchor">{
      <anchor>{
        <doc>{ $t }</doc>,
        <path>{jj:build-Path($e)}</path>
      }</anchor>
    }</info>,
    <info name="user">{string($e/@user)}</info>,
    <info name="room">{string($e/@room)}</info>,
    <info name="content">{$e/chat/text/text()}</info>
  }</item>
}</display>
```

Note the use of the external module "jjutils.xq", which contains some utility functions (created by JJ Girardot). In particular, here we use the "build-Path" function which constructs the path of an element in an xml document.

Transformations into other formats

Tatiana is capable of reading and displaying display files that do not have the compulsory "time" and "src-anchor" properties (in other words, whose items are not proper events). This can be useful for presenting calculated data such as word counts, contingency graphs, etc.

Tatiana can also export to other xml formats (there is currently no export mechanism. You create a filter whose result will "fail" to be displayed, but which will nevertheless have undergone the scripted transformation. The resulting file can be found in `TatianaWorkspace/Tatiana/cache`).

While XQuery has no inbuilt mechanisms for exporting to plain text, the XQuery engine used by Tatiana has a serialization function which can be used to generate plain text or html.

Filters

Examples of filters

```
<?xml version='1.0' encoding='UTF-8'?>
<display>
  <item>
        <info name="description">GraphML file</info>
        <info name="result">0</info>
  </item>
  <item>
    <info name="filter">select-graphml</info>
    <info name="param">0</info>
  </item>
</display>
```

This filter imports data from the GraphML file format. It will ask the user for a "GraphML file". This file will then be passed as a parameter to the *select-graphml* script (found in the file `TatianaWorkspace/Tatiana/scripts/select-graphml.xq`).

```
<?xml version='1.0' encoding='UTF-8'?>
<display>
  <item>
        <info name="description">Coffee Trace File</info>
        <info name="result">0</info>
  </item>
  <item>
        <info name="description">Group name</info>
        <info name="type">String</info>
        <info name="default">group1</info>
        <info name="result">2</info>
  </item>
  <item>
    <info name="filter">select-coffee-graphicaltool</info>
    <info name="param">0</info>
    <info name="result">1</info>
  </item>
  <item>
    <info name="filter">filter-by-info</info>
    <info name="param">1</info>
    <info name="param" paramtype="value">group</info>
    <info name="param">2</info>
  </item>
  <item>
```

```
        <info name="layout">graphicaltoolcoffee.xml</info>
  </item>
</display>
```

This filter imports data from the graphical tool portion of a Coffee Trace file, restricting that data to one group only. It asks the user to select a Coffee Trace File and to enter the name of a group (this field will default to "group1"). The Coffee trace file is passed as a parameter to the script *select-coffee-graphicaltool*. The result of this transformation is passed, along with the value "group" and the name of the group entered by the user to the script *filter-by-info* (thus allowing only the items where the group facet has the value entered by the user). The result of this is laid out with the `graphicaltoolcoffee.xml` layout (which is to be found in the folder `TatianaWorkspace/Tatiana/layouts`).

More examples can be found in the Tatiana distribution, in the filters folder.

Description

Tatiana filter files are a subset of Tatiana "display" files. They are composed of a series of item, which are in turn composed of a series of named info.

Tatiana filter files handle variables through a single array. Variable "names" are therefore restricted integers. In this document, the word "variable" will therefore always refer to such an integer.

Tatiana filter files are separated in four parts:

`input*` -- describes the inputs and assigns each to a variable

`transformation*` -- describes the transformations, their parameters (variables that have previously been assigned) and assigns the output to a variable

`output` -- describes the last transformation

`layout` -- describes the layout file to be used for the result of this filter

Each `input` is an item whose first info is named "description". The content of this info is a free text description of what this input should refer to. This is followed by an info named "result" which contains the variable in which to store the filename of this input. Inputs may optionally contain an info named "type". If the type is set to "String", the user will be presented with a text input box. Otherwise, the input is assumed to be a filename and the user is presented with a dropdown list of possible files. Inputs may optionally contain an info named "default". This will present the user with a default value.

Each `transformation` is an item whose first info is named "filter". The content of this info is the name of an XQuery file, minus the XQuery extension and residing in the scripts/ directory. This is followed by one or more info named "param" describing the variables to be used as parameters to this transformation. The final info is named "result" and contains the variable in which to store the filename of the output of this transformation. By setting the "paramtype" on a "param" to "value", this parameter is now no longer read from a variable, but is passed as a value. The `output` is a `transformation` containing no "result" info. It describes the final transformation of the filter. The `layout` is an item whose first info is named "layout". This info contains the relative path to the layoutfile to be used. In an ideal world, the `layout` would not belong in this kind of file. In the future, it will be described separately and is currently included for convenience.

Appendix VI – Tatiana user manual

Introduction

What is Tatiana and who is it aimed at?

TATIANA, the Trace Analysis Tool for Interaction ANAlysts, is a set of software that helps educational and congnitive scientists in analysing corpuses of interaction data. Tatiana is a tool for manipulating and creating analysis artefacts. It was built for researchers, based on how they constructed and conducted previous analyses in the past. Tatiana was designed during the europeen project LEAD[25] so researchers could analyse data collected by the CSCW tool CoFFEE[26] (developed during the same project)

What can I do with Tatiana?

As an educational or cognitive scientist, you can analyse corpuses of interaction data: find interesting moments (search, filter, combination of views), understand what happened (synchronisation of different media, highlighting, "brush and link", replay) and perform the actual analysis (categorisation, annotation, transcription, description, explanation).

Setting Things up

Hardware and software requirements

Hardware

[25] http://www.lead2learning.org
[26] http://CoFFEE-Soft.com/

As the Tatiana software itself is a Java application, it requires a Java Virtual Machine (version 1.5 or higher) to run. Thus, it will run on every personal computer powerful enough to support Java 1.5 or higher, regardless of the operating system (see below).

One of Tatiana's interesting assets is to be able to control multiple external sources (videos, audio samples, CSW software replayers, etc), and thus if you plan on manipulating multiple sources at the same time, a recent computer may be necessary. We suggest the equivalent of a Pentium IV with at least 500Mb of RAM.

Software

- A supported operating System:
    1. Mac OS X 10.4 or newer (no recent enough version of Java is available for earlier versions of Mac OS X);
    2. Windows 2000/XP (not tested with Windows Vista yet);
    3. A Linux distribution with a 2.6 kernel.
- Java 1.5 or newer:
    4. Java can be downloaded for free at the following addresses:
        - For Windows: http://java.com
        - For Macintosh: http://apple.com/java
        - For Linux: http://java.sun.com/linux
- Quicktime:
    5. If you plan on using videos inside of Tatiana, you must have quicktime installed on your computer. If you system can read quicktime videos, then Tatiana will be able to too. If you need to install quicktime it can be found at the following address:
        6. http://www.apple.com/quicktime/download
- To enhance the usefulness of Tatiana, several tools can be installed to extend Tatiana's possibilities (related tools needed to perform analysis, depending on what you need to analyse):
    7. The CoFFEE replayer[27];
    8. The DREW replayer[28];

[27] http://lead2learning.org/
[28] http://lead.emse.fr/ or http://Tatiana.emse.fr

Downloading Tatiana

To be able to use Tatiana, you first need to download and install the software. To do so take the following steps:

9. Go to the http://lead.emse.fr or http://tatiana.emse.fr website;
10. Click on the Download section on the left hand side of the page;
11. Scroll down, then read carefully and accept the CeCILL license.
12. Download the latest version of Tatiana that corresponds to your operating system.

Installing Tatiana and related tools

Tatiana

If you are using windows, Tatiana can be downloaded in two forms:

- Tatiana-setup.exe: to install simply double-click to launch setup procedure, and follow on screen indications.
- Tatiana.zip: to install, unzip the archive to the desired directory. If you need a zip archiver, you can use the free 7 zip downloadable here www.7-zip.org

Macintosh versions are distributed in a zip package too; unzip to desired directory.

CoFFEE

The CoFFEE Replayer is now bundled with the CoFFEE software. To install the CoFFEE Replayer you just have to select the right option while installing CoFEE. More information can be found in the CoFFEE documentation, available at the LEAD website[29]. By default the CoFFEE Replayer is a standalone piece of software that can replay CoFFEE trace files. To enable it to accept remote control commands from Tatiana, you have to do the following:

- Open the directory where you installed CoFFEE (by default C:\Program Files\CoFFEE if your using Windows)
- Locate the Replayer directory
- Open the Replayer.ini file that is in that directory
- Delete the parameter `-standalone`

[29] www.lead2learning.org

When you launch the CoFFEE Replayer, you won't have the remote control any more, and the CoFFEE replayer will await commands from Tatiana.

Drew

To install the Drew replayer, simply unzip it to the desired directory. More information on how to use the Drew replayer with Tatiana can be found in the section "Using the Drew replayer".

Tatiana

Basic concepts

Tatiana is a tool for manipulating and creating analysis artefacts. At the moment, Tatiana only takes into account the kind of artefacts we call *replayables*. These are artefacts that are time driven and can be "replayed". All replayables in Tatiana can be:

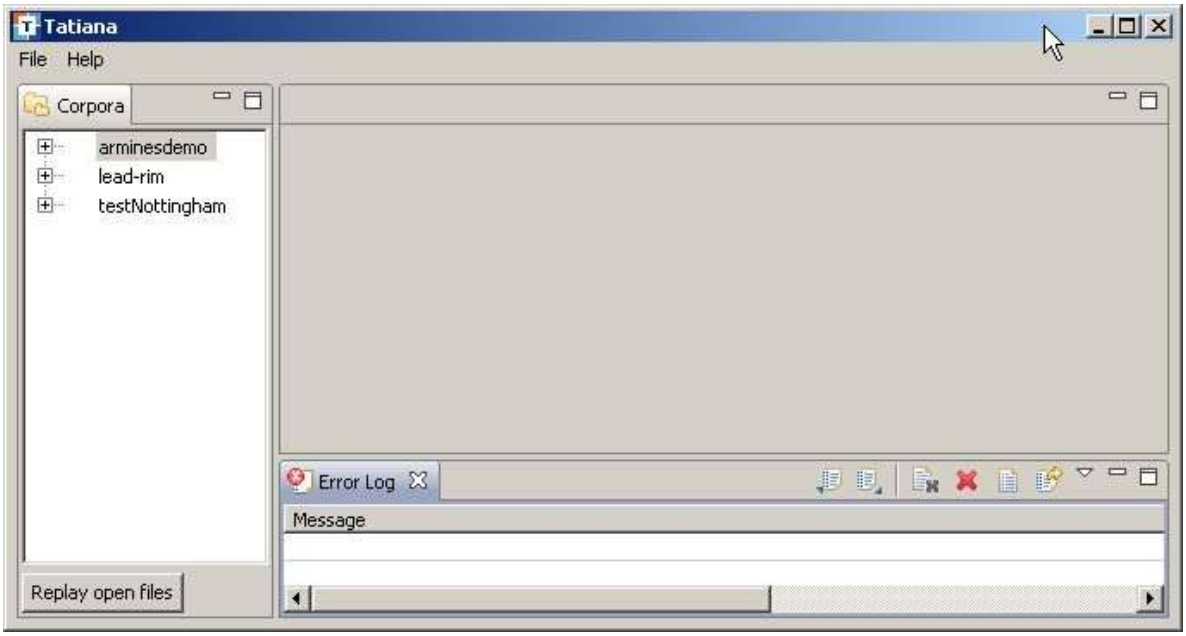
- Viewed in tabular form
- Viewed in graphical form
- Synchronised with other replayables
- Annotated and Categorised
- Exported to Excel

Other objects that Tatiana understands are *trace files* (either video or xml source files), *external replayers* and *analyses*.

At the moment, there are no other kinds of objects in Tatiana.

Launching Tatiana

Depending on the Tatiana package you downloaded, you may not have an icon/shortcut to launch Tatiana. In this case, navigate (using a file browser) to the Tatiana directory, and launch either the Tatiana.exe (windows) or Tatiana (Macintosh) file.



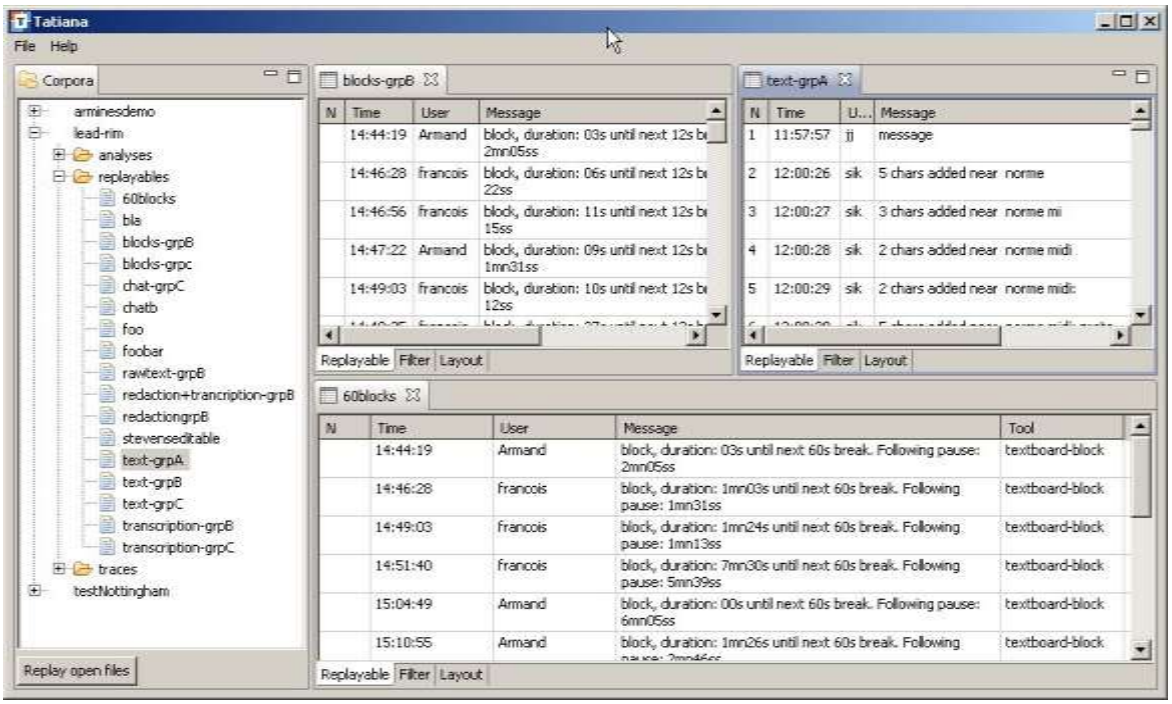
9 - Tatiana upon launch

Please note that captions are made with the windows version. Look and feel may vary depending on the operating system you are using.

If this is the first time you are launching Tatiana, please refer to the section First Steps with Tatiana.

The Tatiana window

Tatiana's window has two main areas. The area on the left displays all the corpora and files that Tatiana is currently managing. The area on the right is used to display all the replayables that are currently being viewed. These replayables appear in tabs. You can drag a tab towards the top, bottom, left or right to create a split pane in order to view several replayables at once.

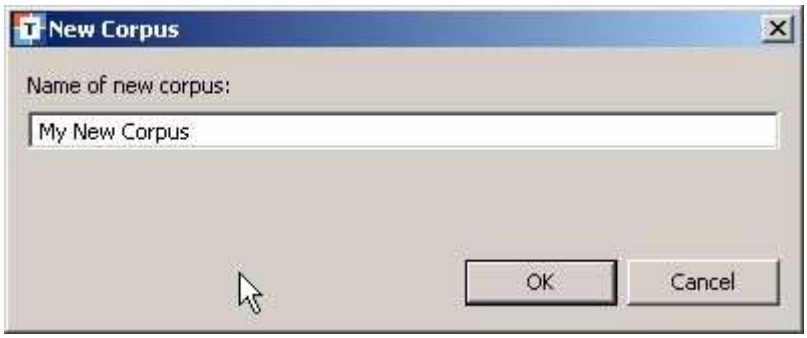


10 – Several replayers opened and view at the same time

Using Tatiana

Create a corpus

A corpus is a consistent set of trace files, replayables and analyses. The choice of what "consistent" means is entirely up to the researcher. To create a new corpus, go to File -> New Corpus.

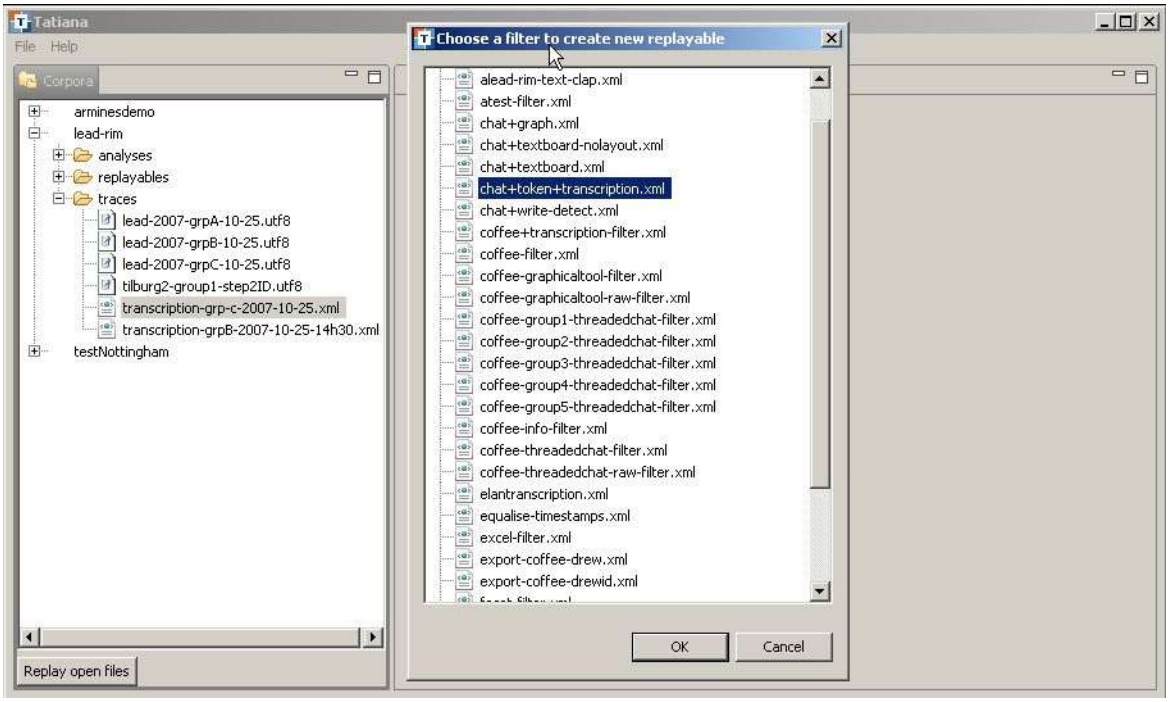


11 - New corpus dialog box

When working with several corpuses in Tatiana, most actions you do will be done on the active corpus (the one that is selected).

For example below the *chat+token+transcription.xml* filter will be applied to the selected file (*transcription-grp-c-2007-10-25.xml*) from the lead-rim corpus. Any action made on a corpus

(for example adding a new file) will be made on the selected corpus (or the corpus containing the selected file).



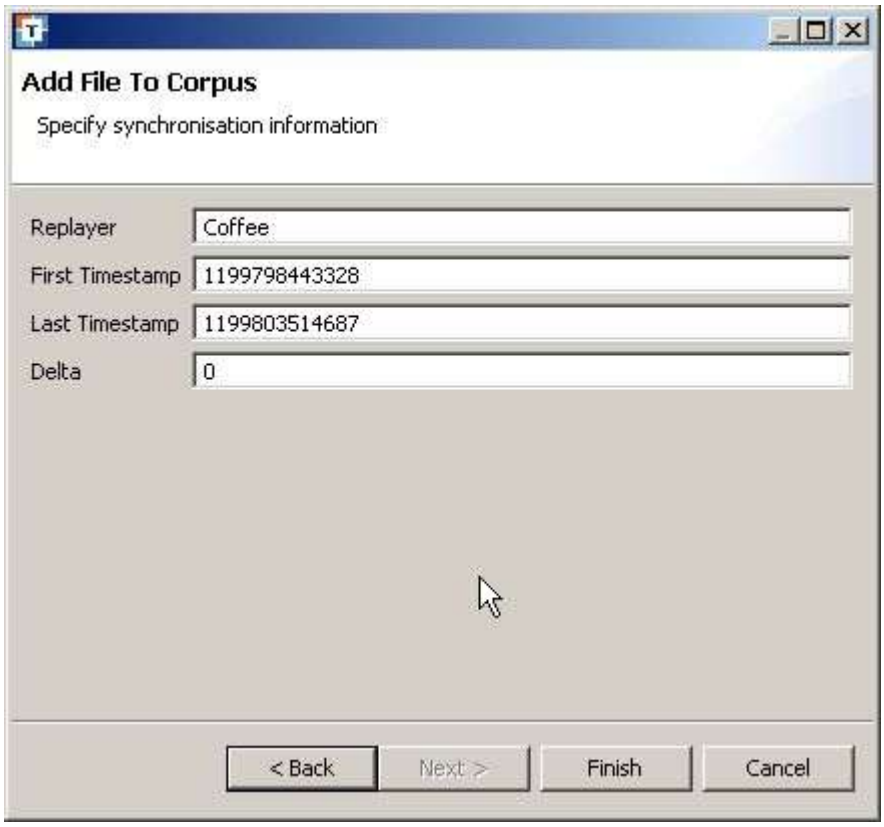
12 - Applying a filter to a selected file

Adding files to a corpus

Basic procedure

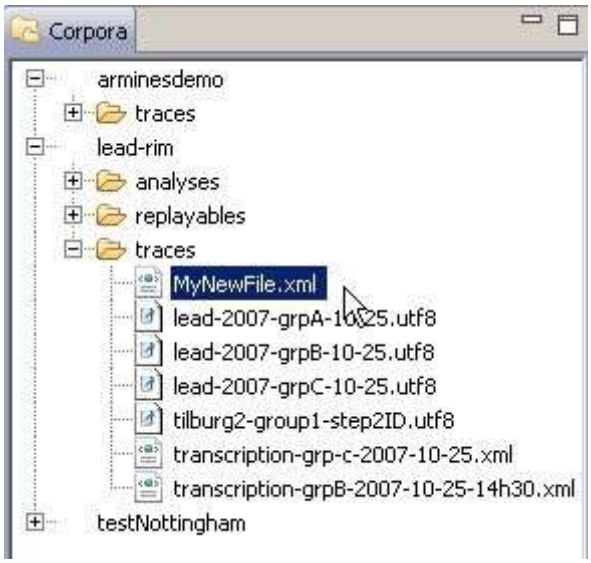
To add a file to a corpus go to `File-> Add File to Corpus`.

When adding trace files to a corpus, you will need to specify some information to allow Tatiana to synchronise the various data sources. When you select the Add File to Corpus menu, you will be asked to locate on your computer the file you would like to add. Choose a file and then click Next. This will bring up a configuration screen where information is needed to align time wise the file you are adding to your corpus.



13 - Synchronisation information when adding a new file

For CoFFEE and DREW trace files, Tatiana will automatically calculate the synchronisation information (Tatiana extracts this information from the trace files). For other types of files (transcriptions from other software, videos, etc), you will need to calculate this information yourself and specify it here. To finish the procedure of adding a file, click on "finish". Your new file will appear on the left in your corpus.



14 - New file added to current corpus

If you are adding a large file (like a video file), please be patient, because the file will be copied from it's location into the directory that contains your corpus in Tatiana.

If you make a mistake while filling in the timestamps, or if you find that your media (video for example) is slightly out of sync, you can change them like this: it is currently not possible to edit the synchronisation information in Tatiana. However, importing again a file that is already in Tatiana will allow you to specify the synchronisation information again without copying the file a second time. So go to `File->Add file to Corpus`, and then select your tracefile in your Tatiana workspace (`<yourcorpus>/tracefiles/filename`). See chapter *Organisation of files in the file system* for more information on the Tatiana workspace.

Synchronisation timestamps

While we expect to provide more simple ways to synchronize the media in future versions of Tatiana, users of the current version are encouraged to use the following approach.

We will briefly present how our experiments where conducted, and what we did to ensure easy import of media into Tatiana for later analysis.

Our experiments where done in a single room with a teacher and students using computers running collaborative software, and with video and audio recordings. Most of the time the following procedure was used (and was found to be appropriate): the teacher typed the word 'clap' in the chat window of the CSCW software. Then while saying out loud "now is the initial clap", he would hit the "enter" key with the right hand (sending the 'clap' message in the chat), and simultaneously hitting the table with the left, all this under the eye of the camera. This way all timed media we will use later has this special clap event easily recognisable in its recording:

- The trace file produced by the CSCW software has the word 'clap' appear in the chat tool at a specific timestamp
- In the video, it is easy to find at which point the left hand hits the table
- On the audio recordings, the clap sound made on the table with the right hand is easily recognisable

This allows us to synchronise in Tatiana the video, the (independent) audio recording(s), and the CSCW tool.

Now, let's consider the various aspects of these different media:

- audio and video are continuous media, that start at a relative time "0". Most of the time, we can express a position in the audio or video record as a relative time expressed in seconds with a decimal part, like 17.291 seconds for example (or, in some cases, in seconds and frames, like 17 seconds and 7 frames, which is equivalent for a 24 frames/second video).
- the computed mediated traces correspond to discrete events (one of such events is the "clap" message in the chat), which are timestamped with an "absolute date". For example 11h33m22s:12/01/2008 or the equivalent as a unix timestamp: 1200137602.

Of course, the various recordings are not necessarily started simultaneously, and there is usually a few seconds, sometimes a few minutes lag between the beginnings of the actual recordings. During an experiment, one may check the software is running, then start the video recording, and then start the audio recordings, before really starting the session with the clap procedure explained above. The difficulty is to obtain the most precise possible value of this lag. We use the following procedure.

In Tatiana, we import the video with the menu File->Add file to Corpus. We browse the files on the computer, select the video (with a typical suffix like ".mov" or ".mpg", or anything that Quicktime can open) then click on "next", which opens a window with three fields:

- (F) first time stamp
- (L) last time stamp
- (D) delta

The values here should be expressed in milliseconds. You need to fill up all three fields. What is expected from these values is the following properties:

- F+D is the actual value of the timestamp of the first event in the file. The timestamp of a temporal event in Tatiana is, according to the classical UNIX convention, a number representing the number of milliseconds passed since January 1st, 1970. If you need to convert a time/date to unix style date in milliseconds, you can use the following website: http://www.onlineconversion.com/unix_time.htm For a video the first event is defined as the first frame of the video. For an audio recording the first event will be the first sample of the digital audio recording; it will be the first intervention for a transcription, etc.
- L+D is the actual value of the time stamp of the last event in the file, with the same conventions as described above. L-F is therefore the duration in milliseconds associated to the medium.

As can be seen, two values only are needed. Using three fields provides a little flexibility that simplifies the description of the medium, as will be seen in the following examples.

Case of a CSCW tool such as DREW

The values F and L are directly available from the file, and correspond to actual UNIX time stamps recorded by the tool. Look for the first and last event in the file, the corresponding "time" information will provide the values of F and L. Since F and L are appropriate values, the field D should be set to 0.

```
(machine de turing?)
surface 2D</text></textboard></event>
<event mark="http test step 1" user="francois" room="Hall">
<time><date>1193316548474</date></time><textboard><text
lang="en" caret="91">interface graphique
utilisation scientifique et artistique
(machine de turing?)
surface 2D </text></textboard></event>
```

15 - Extract of a Drew tracefile

Case of an audio or video recording

Now, let's imagine you have a 33 minutes and 8 seconds long video made during an experiment as described above. You play the recording, waiting to see the teacher hitting the table for the "clap". Evaluate as precisely as possible the position of this "clap" in milliseconds since the beginning of the record. This is a relative value that we will call "R". So if the "clap" appears 12 seconds and an half after the beginning of the record, R is 12500. Now, look for the event corresponding to the "clap" in the CSCW traces. The associated "TIME" is for example the number 1200582784 (which corresponds to January 17, 2008, at 15h13:04). Let's call it "T", actual time of the clap. The actual value of the beginning of the recording is therefore 1200582784 – 12500, which is 1200570284, and this value can be put in field D. Field F is therefore 0, and field L should be set to the actual value of the duration of the recording, in this case (33x60+8)**x**1000, or 1988000**.**

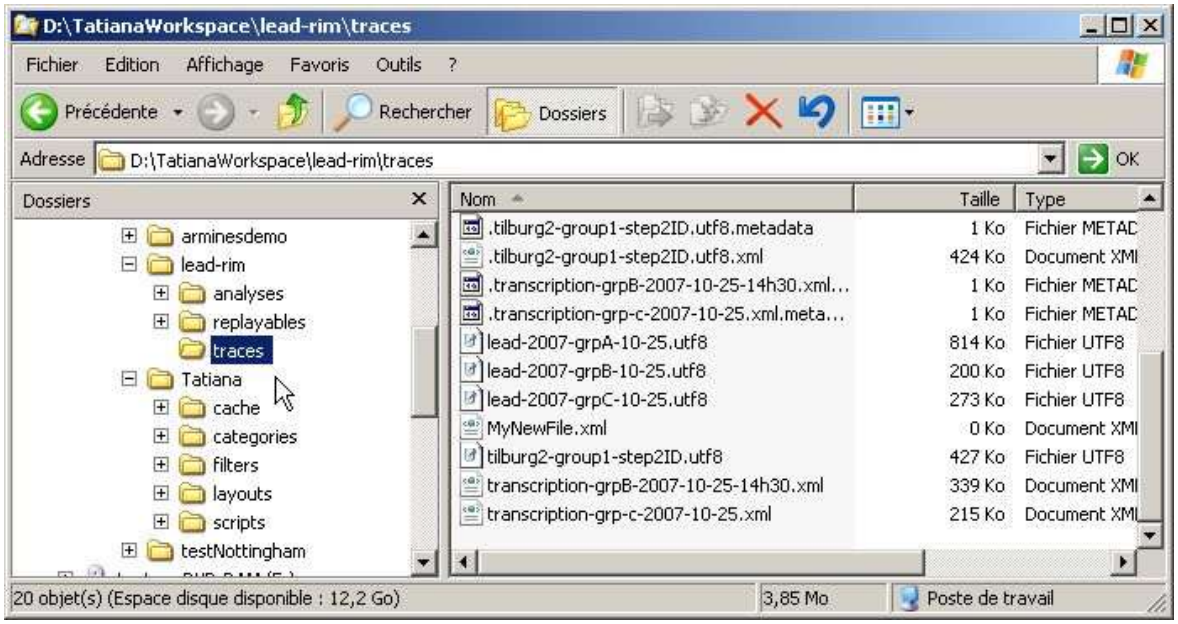
As can be seen from these examples, any set of F, L and D values that give for F+D and L+D the real beginning and ending time stamps for the medium is legitimate. However, for video files, F must be set to 0 and L to the length of the video in milliseconds.

Organisation of files in the filesystem

Now we have gone through the first steps of using Tatiana (creating a corpus and adding files to it), we will explain how these files are handled by Tatiana.

In your Tatiana directory (where you installed Tatiana), you will have a directory called "Tatiana Workspace". Inside this directory, you will find one directory for each of your corpora, some other miscellaneous directories and a directory called Tatiana. Inside the Tatiana directory can be found the scripts and filters which are used to import and transform data.

Inside each of your Corpus directories, there may be a tracefiles directory into which Tatiana will automatically copy the files you added to the corpus, a replayables directory where Tatiana will automatically save your replayables and an analyses directory where Tatiana will automatically save your analyses.

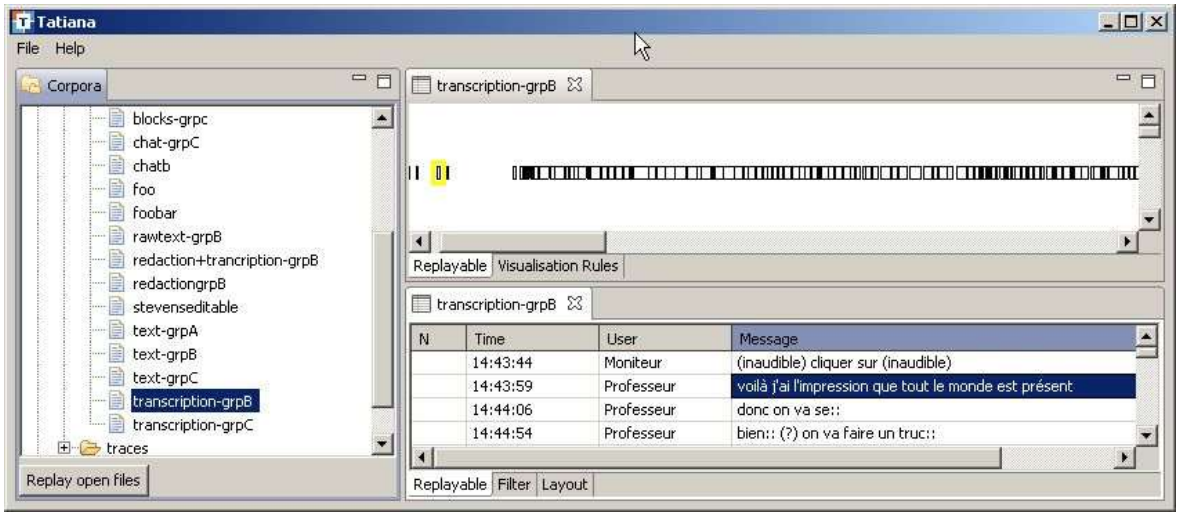


16 - Tatiana's Workspace

Replayables

Tatiana uses artefacts we call *replayables*. These are artefacts that are time driven and can be "replayed". All replayables in Tatiana can be:

- Viewed in tabular form
- Viewed in graphical form
- Synchronised with other replayables
- Annotated and Categorised
- Exported to Excel



17 - A replayable in tabular and graphical form

Tatiana understands two kinds of replayables:

- Those that are automatically created by means of a filter
- Those that are created by hand by the analyst

These replayables are only distinguished in the tabular view. They are treated identically in all other cases (analysis, synchronisation, source of new filters, graphical visualisation and export). Replayables can (and should) be saved for later reuse.

We will now briefly present the first kind of replayable, but will need to present the notion of synchronisation before being able to present the second.

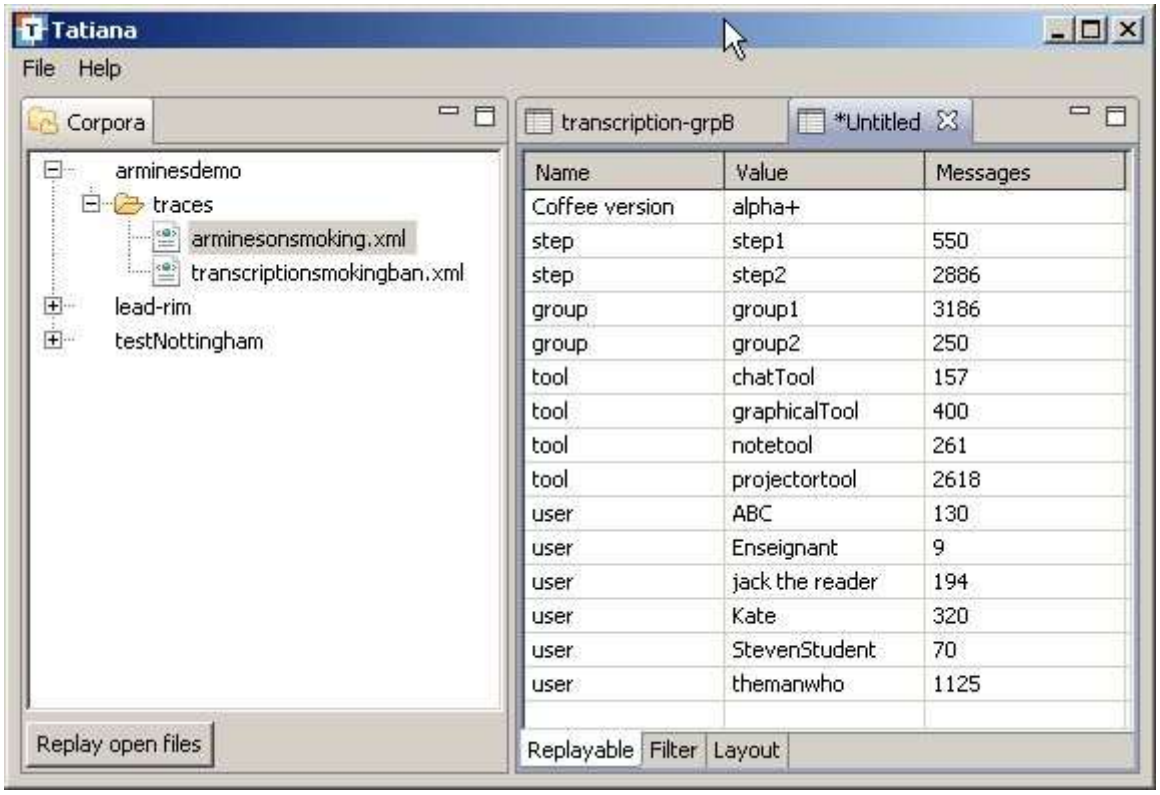
Replayables from filters

To create a new replayable, goto `File -> New Replayable -> Replayable from filter`. You will be asked to choose a filter (or transformation).

18 - A list of filters

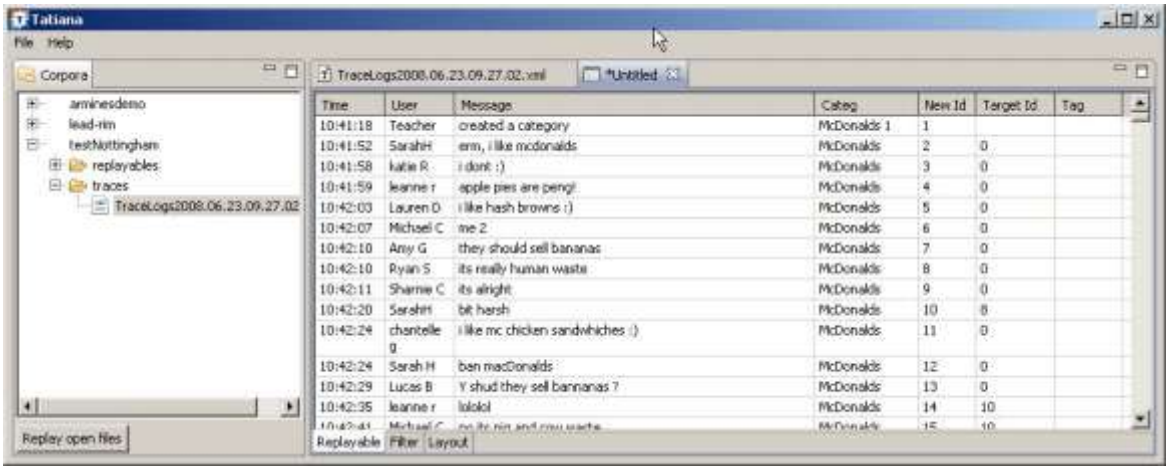
Filters act like black boxes with inputs and an output. You can use any replayable or trace file as an input to a filter. This input or combination of inputs will then undergo the specified transformation. Some examples of filters include:

- coffee-info-filter.xml:
  - this filter goes through a CoFFEE trace file, and extracts information from it
  - input: a CoFFEE trace file
  - ouput: shows a summary report with statistical information (number of messages, numbers of groups, etc) from a CoFFEE trace file.



19 - Execution of the information filter on a CoFFEE tracefile

- coffee-threadedchat-filter.xml:
  - this is the basic filter to extract threaded chat events from a coffee trace file
  - input:
    - the name of a group (as defined in CoFFEE)
    - a CoFFEE trace file
  - output: produces a replayable containing the threaded chat events for the specified group.



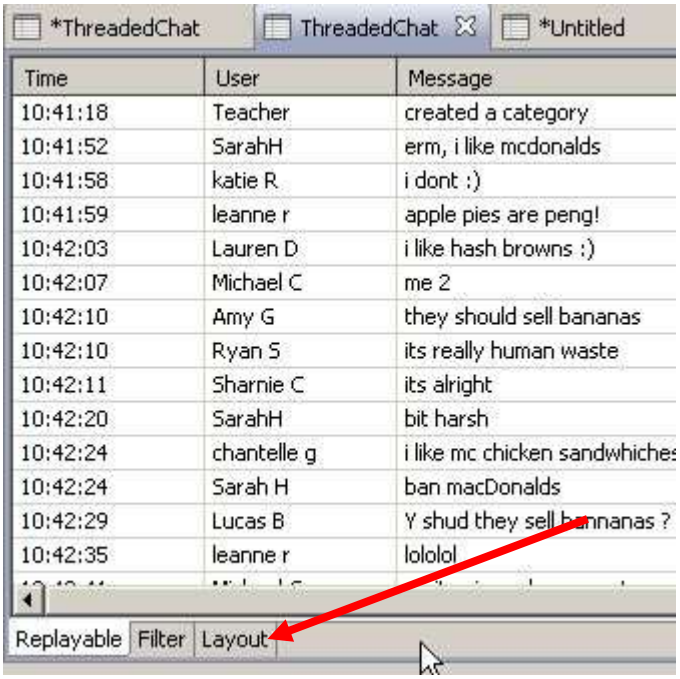
20 - Threaded Chat extracted from a CoFFEE tracefile

- coffee-graphicaltool-filter.xml:

        - this is the filter that extracts events from the CoFFEE graphical tool
        - input:
            - the name of a group (as defined in CoFFEE)
            - a CoFFEE trace file
        - output: produces a replayable containing the events from the graphical tool for the specified group.
    - facet-filter.xml:
        - this filter is to prune out unwanted information from another replayable
        - input:
            - a replayable
            - a facet (event property) name
            - a value
        - output: it filters out all events which don't have that value for that facet.
        - example: try applying it to a replayable which was created with the coffee-threaded-chat-filter, using facet name "step" and facet value "step1".

To apply a filter, choose the appropriate inputs and click "run filter".

Notice that this kind of replayable, when viewed in tabular form has three tabs at the bottom.

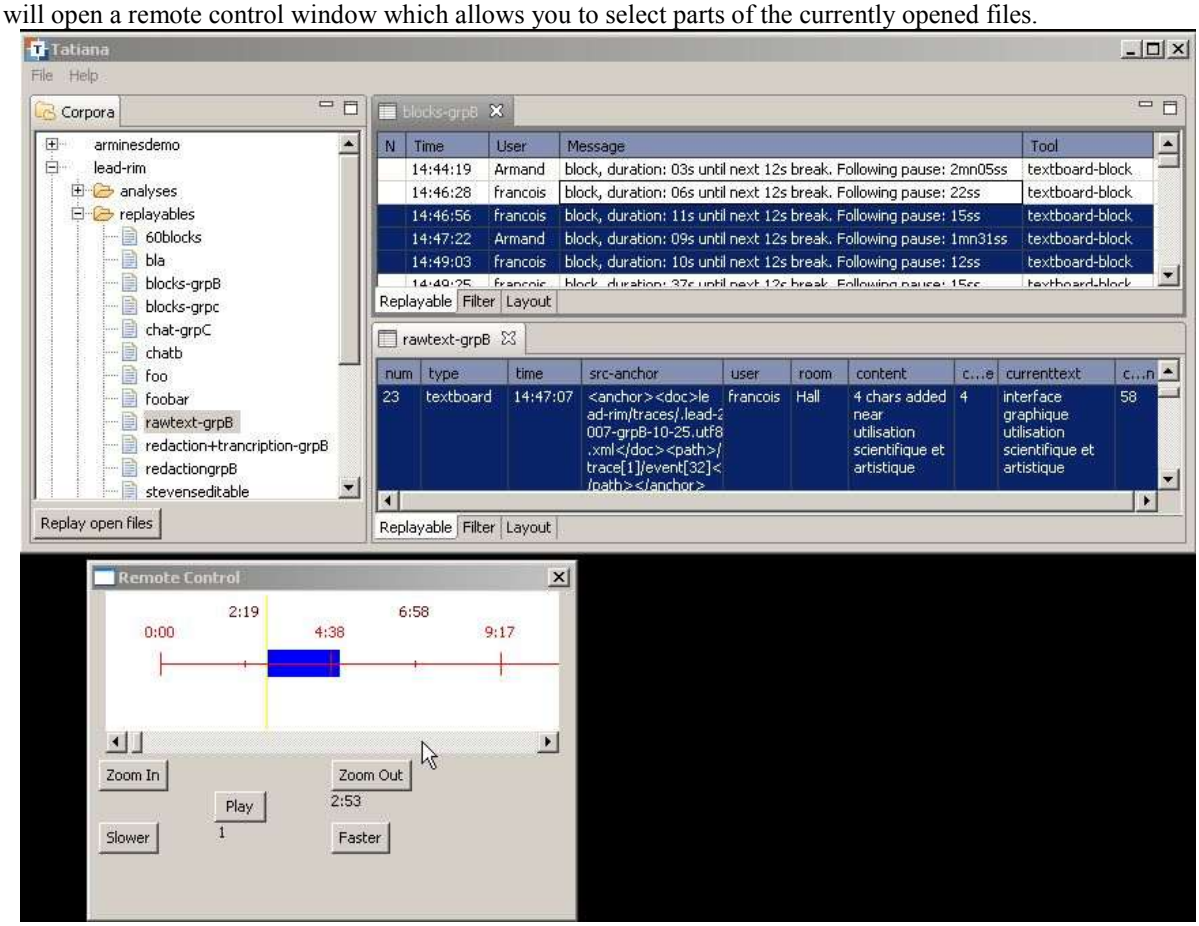


21 - the 3 tabs of a replayable

The first tab of a replayable shows the result of the filter, the second shows the parameters that where used when the replayable was made. To change the result of a filter, simply change the parameters on the second tab, and rerun it. The third tab is not very useful at the moment and can be disregarded (it contains for the moment the name of the layout associated to the replayable; more information on layouts can be found in the Tatiana Technical Documentation).

Synchronisation

While in later versions of Tatiana, this will be done by automatically, at the moment, you must manually click on "Replay open files" to get a synchronized replay of all the replayables and trace files that are open. Doing this will open a remote control window which allows you to select parts of the currently opened files.



22 -Synchronised replayables and remote control

Note: by "open" we mean objects that you have open in the right side area of Tatiana (the ones you have double clicked on). For unfiltered tracefiles and audio/video files this will, at first, only show the synchronisation information that has been entered.

For replayables, this means that events will be highlighted when they are "current" and selecting an event will hightlight matching events in all other replayables.

For video files, provided you have Quicktime installed on your machine, they will automatically be opened when you click "Replay open files".

If you want trace files to be replayed by their corresponding replayer, you must have a corresponding external replayer already opened (such as the DREW or CoFFEE replayer). Please consult the CoFFEE documentation to see how to ensure that the CoFFEE replayer can be piloted by Tatiana).

Termination of the synchronisation will happen if you close the remote control or if you close one of the current replayables. If you open a new replayable, it won't be included in the current synchronisation. If you terminate synchronisation which uses an external replayer, you must re-launch this replayer before synchronising again.

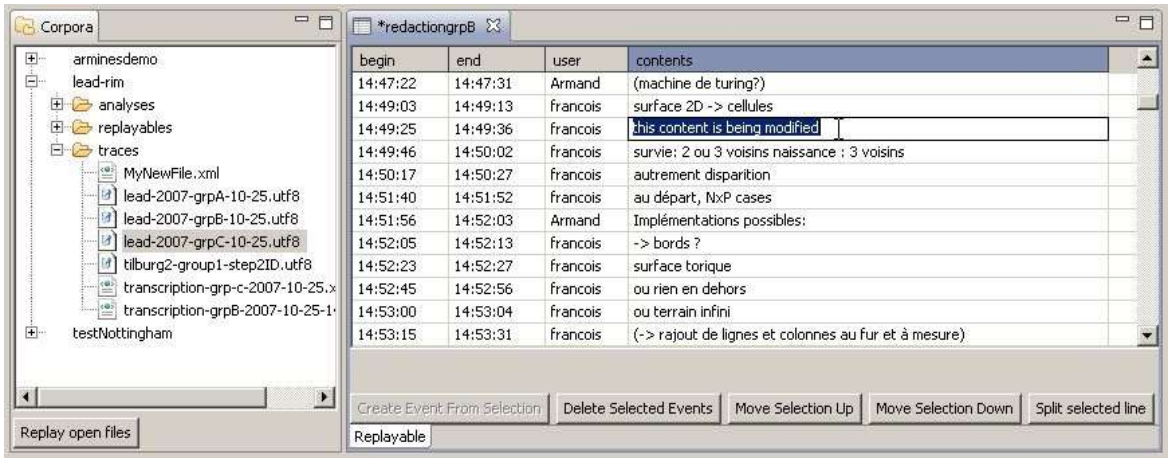
Replayables created by hand

Filtered replayables are useful when there exists some kind of automatic transformation to create the artefact you want from those you already have. Sometimes however, such as in the case of transcription, manual grouping of events, creating new events, reordering of events, deletion of events, etc, you need manual control. We have chosen to distinguish completely between the automatic kind of replayable and the manual kind.

There are two ways of creating a replayable that can be edited:

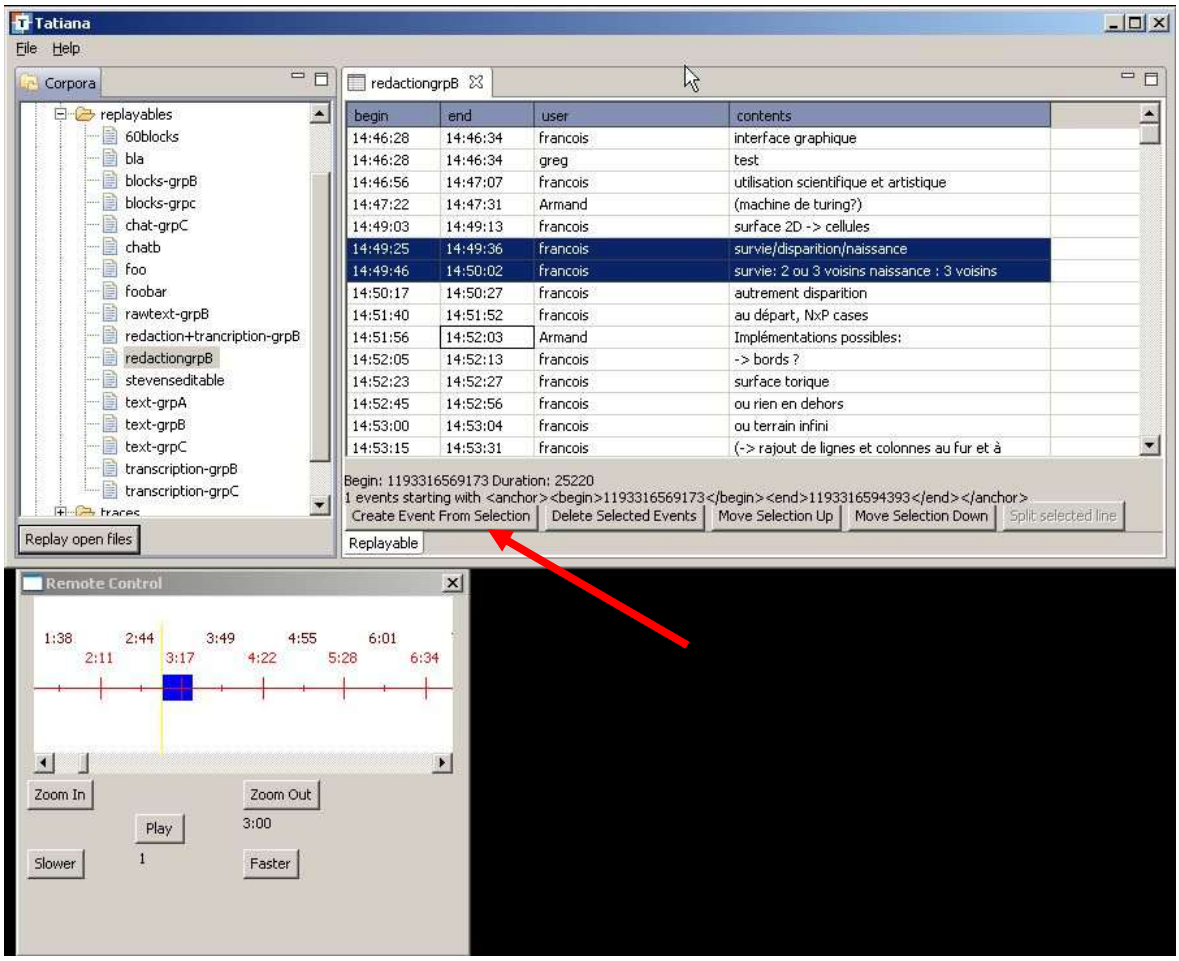
- File -> New Replayable -> Replayable from scratch...
- Open an existing replayable and go to File -> Save an editable copy...

You can then delete, move and edit events on this replayable. To edit an event, simply clic on it, to start editing. To delete an event or move it up or down, select it, then press the corresponding button (the buttons are located at the bottom of the tab containing the replayable). You can also duplicate an event, by clicking on the `Split selected Line` button. This will split the event into two identical events, that you can then modify as you want.



23 - modifying a replayable

Once it has been synchronised with other replayables (or with a video for example), you can select several events (or a time period in the remote control) and click "Create Event from Selection". This will insert a new event at the appropriate place.



24 - Creating an event from a selection

Analyses

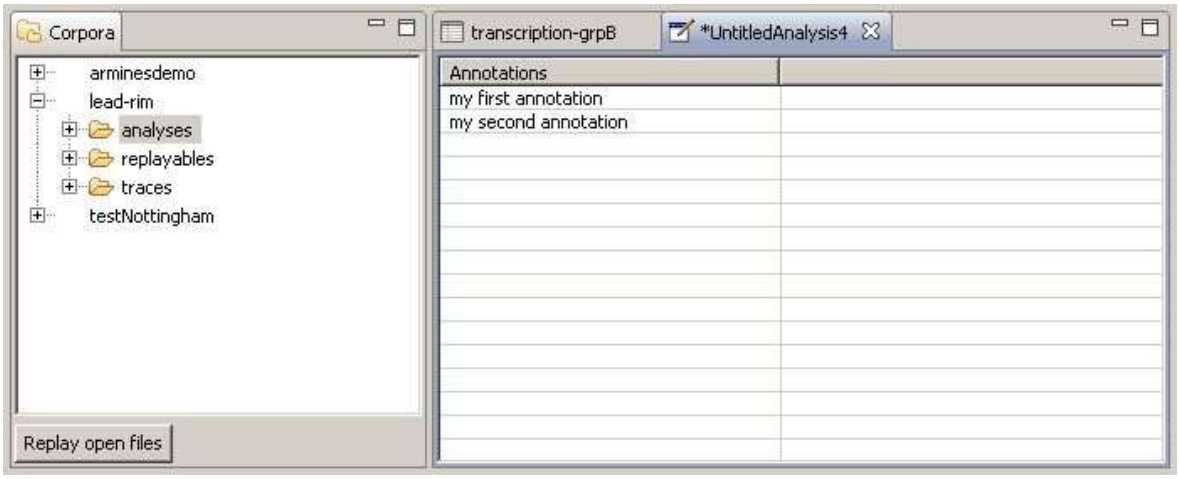
Analyses in Tatiana add a new column to all open replayables in the corpus. The cells in this column are free text in the case of annotation analyses and a dropdown menu for categorisation analyses. You can have as many simultaneous analyses open as you like.

There are four ways of creating a new analysis:

- File -> New Analysis -> Annotation
- File -> New Analysis -> Categorisation (from scratch)
- File -> New Analysis -> Categorisation from file...
- File -> New Analysis -> Graph

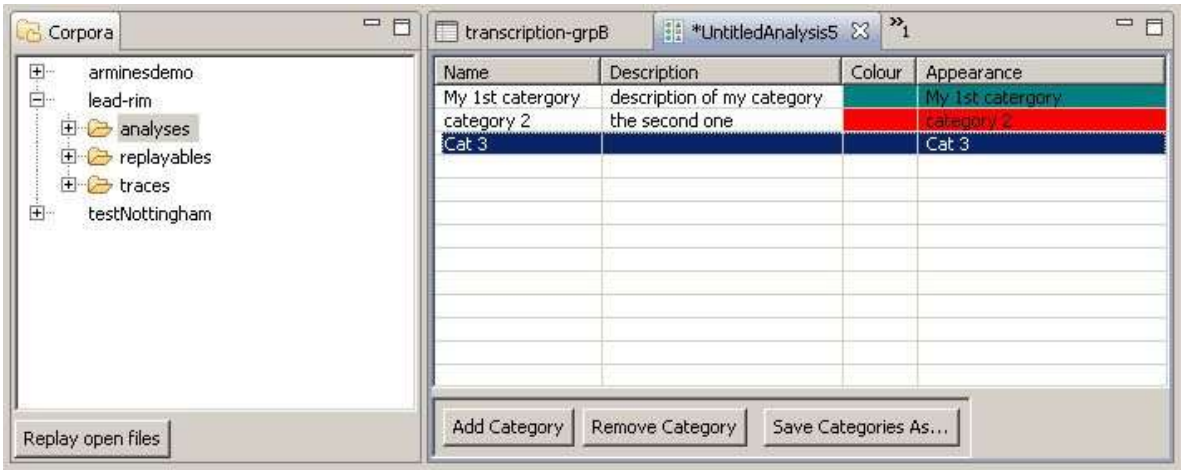
The first method adds an extra column to the replayable, so you can add annotation to existing events. A new tab if created to containing all annotations you add in the extra column.

25 - Annotating to analyse (in extra column)



26 - List of all annotation made in an analysis

The second method gives you a blank slate with no pre-existing categories. The researcher can define categories when he wants by clicking on the `Add Category` button. He can then add a description, and choose a color to assign to that category.



27 - Defining categories for an analysis

The third uses categories from an existing set. When selecting this option, Tatiana will ask the user to search for a previously saved category list, and load the categories for the analysis.

The tab which opens when you create a new analysis allows you to manage and save your analysis, independently of your replayables. The tab created by an annotation analysis only allows you to save the analysis (by selecting the tab and going `File -> Save`). The tab created by a categorisation analysis additionally allows you to add and delete categories, modify their textual content and modify their colour.

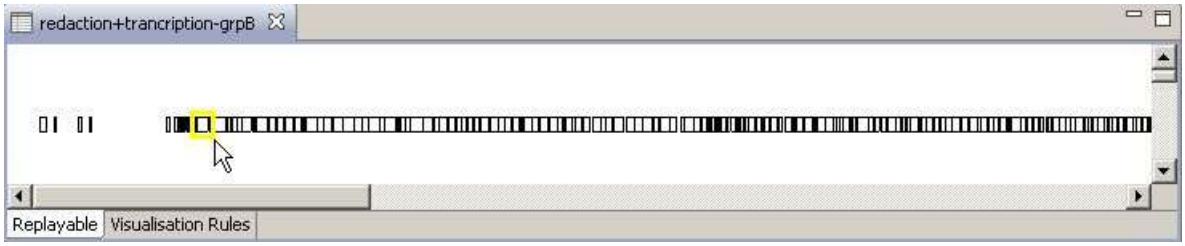
The fourth provides a blank slate in which a graph can be edited. To place an event (represented by a box) in the graph, select *yes* in the dropdown list provided in replayables. To remove it, select *no*. Three different kinds of lines can be drawn between events. Lines between events can be annotated with text and the text in boxes can be modified.

Interaction Score Sheets (graphical visualisations)

Introduction

Tatiana replayables can be visualised as a table or as a "score sheet". This score sheet takes each event and creates a graphical object for it which will be displayed according to rules. The horizontal position of an event is constrained by the time at which the event occurs and the width by the duration of the event. The other properties of the object will be determined through the application of rules according to the other properties of the event. To make this

kind of replayable, open a graphical visualisation by selecting a replayable and go to `File -> View as Score`. Each line or rectangle you can see will match an event.



28 - Graphical visualisation of an replayable

Visualisation rules

You can now select the "visualisation rules" tab. At first you have an empty table with no rules. A rule is composed of the name of a facet, the value that facet should take, the property which should be modified and the value that property should take.

The construction of a rule is to first create a condition by selecting a facet (one of the properties of an event in a Tatiana Replayable) and a value for that facet (or choose "any" if it should be applied regardless of the value for that facet). As a consequence of that condition being met, a property will be set to a certain value.

The semantics of a rule is therefore:

`for each event: if` **`facet`** `is equal to` **`facet value`** `then set` **`property`** `to` **`property value`**

The property value is always evaluated as a number. It can be a simple number, an expression (using operators `*, /, +, -` and brackets) or draw from the value of a facet. A facet's value can be accessed by using the expression `$facet` (if the facet is not numeric, the value will be its position in a list of first occurrences). A facet's colour can be accessed by using the expression `$$facet`.

The different possible properties are:

- **colour:** Colours are decimal equivalents of their hex values. For ease of use, the variables `red blue green` and `grey` are defined. Events that do not have a duration and are not represented as a shape are only one pixel wide and thus only have a border. The colour property will not affect them.
- **border-colour**: set the border colour of the event.
- **absolute position**: set the vertical position of the event
- **relative position**: move the vertical position of the event by so many pixels
- **shape**: sets the shape. The variables `circle, none` and `triangle` have been defined. If an event takes a shape other than none, its width will be identical to its height.

- **height**: sets the height in pixels.
- **visibility**: 0 is false, 1 is true.

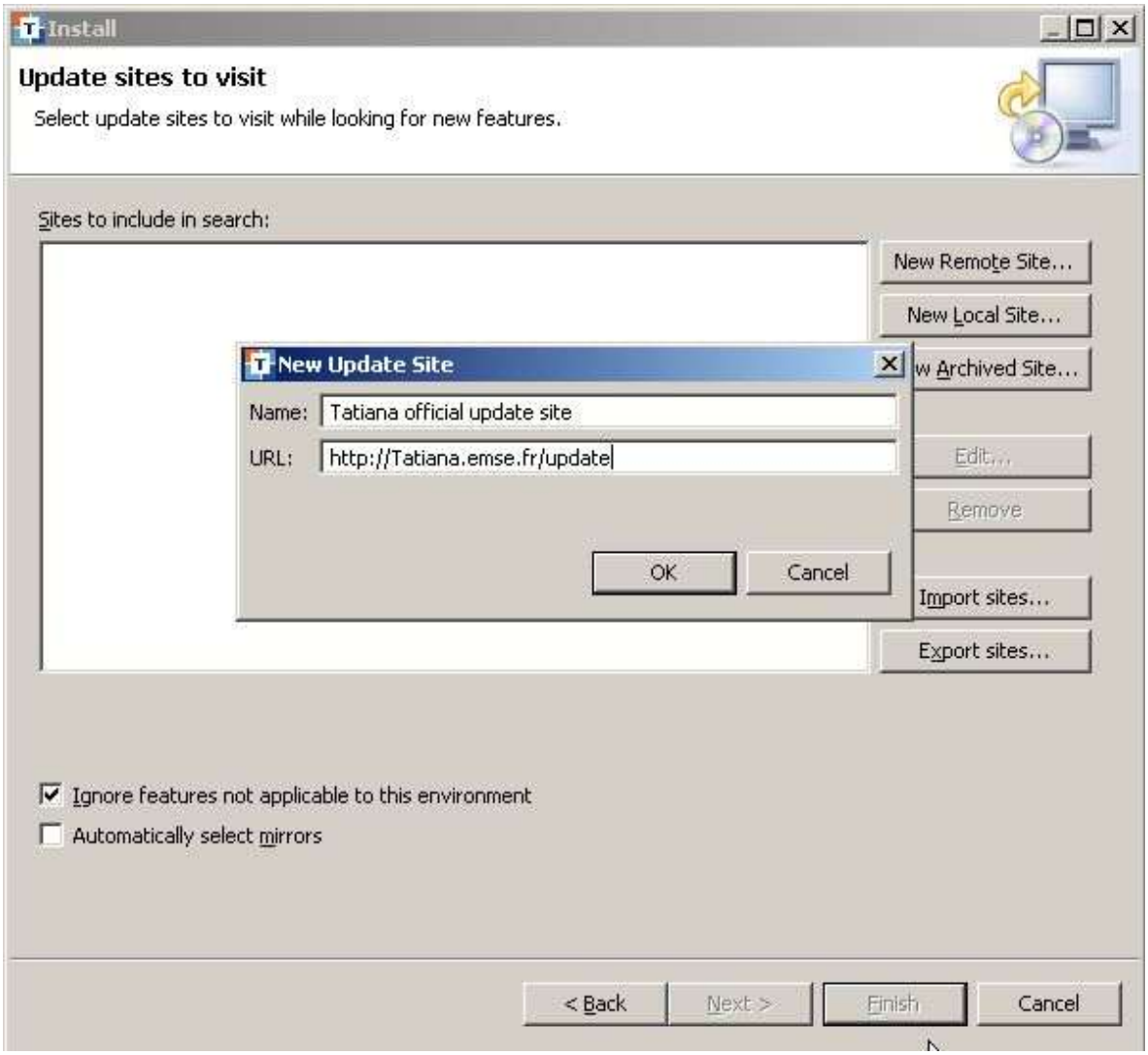
Future implementations

We intend to make the user interface and the model for creating and applying rules much easier to use. We also plan on adding labels and a method for defining links to be drawn between events.

Updating Tatiana

To update Tatiana with a newer version:

- Use the menu and go to Help->Software Updates->Find and Install...
- Select option Search for new features to install, then press Next
- Click on the New remote site button
- Fill in the fields as follows



29 **- Parameters to update Tatiana**

- Click ok, then the Finish button at the bottom
- Tatiana will then search the update site for updates. Available updates will be shown in a tree like view. To update, check the checkbox next to the update wanted, and then press the Next button.
- You have to accept the license to continue and then press the Next button.
- On the last screen where a resume of the update will be shown, press finish to download the update
- Once downloaded press Install to install the update

You might have to restart Tatiana, once the new update is installed.

Appendix VII - Résumé en Français

Les textes suivants proviennent de deux articles soumis à des revues francophones et leur publication (sous forme révisée) devrait se faire dans l'année à venir. Le premier texte traite de la notion de rejouable et correspond plutôt aux trois premiers chapitres de thèse. Le deuxième traite de Tatiana et de nos études de cas et correspond aux deux dernier chapitres de thèse.

Introduction

L'étude socio-cognitive d'interactions humaines médiatisées par ordinateur peut se réaliser au travers d'enregistrements de ces activités interactives, notamment lorsque ces enregistrements ne se limitent pas à des traces informatiques mais incluent aussi d'autres sources de données (vidéo, audio, etc.) (Avouris *et al.*, 2007). La nature complexe de ces données multimodales fait ressentir le besoin et l'utilité d'outils informatiques assistant cette analyse (Cox, 2007)(Dyke *et al.*, 2007)(Fisher et Sanderson, 1996)(Thomas et Cook, 2005), particulièrement dans le cadre de situations d'apprentissage collaboratif (Suthers, 2006). De tels outils sont rendus d'autant plus nécessaires de part les besoins de collaboration et de partage entre différents chercheurs, que ce soit pour valider une analyse (de Wever *et al.*, 2006), répartir la charge de travail (Goodman *et al.*, 2006), combiner les perspicacités de plusieurs chercheurs de domaines différents (Prudhomme *et al.*, 2007), partager des corpus de données (Reffay *et al.*, 2008), ou pour établir un dialogue entre différents points de vues épistémologiques (Suthers, 2006).

Les outils existants dédiés à l'analyse de situations d'apprentissage collaboratif médiatisé par ordinateur (en anglais, *Computer Supported Collaborative Learning*, ou CSCL) sont souvent restreints à une forme d'analyse particulière comme la théorie de l'activité (Fiotakis *et al.*, 2007) ou à une forme de médiatisation particulière comme les échanges langagiers en ligne (Teutsch *et al.*, 2008). Les outils issus d'autres domaines comme l'interaction homme machine (Badre *et al.*, 1995), les sciences du langage (Carletta *et al.*, 2003) ou de l'ethnographie informatique (Greenhalgh *et al.*, 2007) permettent une forme d'analyse plus générique d'interactions mais ne couvrent cependant pas l'ensemble des besoins décrits par les auteurs dans le domaine (Suthers *et al.*, 2007)(Suthers et Medina, 2008).

Une solution à ce problème serait la création d'un format commun permettant à différents outils analytiques d'interopérer (Kahrimanis *et al.*, 2006). Afin d'obtenir des avantages au delà de l'interopérabilité, nous avons proposé la création d'un environnement intégré d'analyse (Dyke *et al.*, 2009a) qui permette de traiter visualisations, transformations et analyses de manière générique. Il existe aussi des travaux de modélisation de la trace informatique d'apprentissage dans une perspective d'utilisation générique de la trace, par exemple pour la ré-ingéniérie des EIAH (Choquet et Iksal, 2007) ou pour la manipuler en vue

d'une représentation subséquente, que ce soit pour l'apprenant, le tuteur ou le chercheur (Settouti *et al.*, 2006). Toutefois ces travaux ne sont pas destinés spécifiquement aux chercheurs et ne couvrent pas l'ensemble de leurs besoins.

Nous souhaitons décrire la démarche analytique du chercheur afin d'en extraire un objet faisant abstraction de toute implémentation qui permette de couvrir une majorité des besoins de manipulation des données issues de situations d'apprentissage médiatisé par ordinateur lors de leur analyse subséquente. La description d'un tel objet permettra :

- de décrire l'ensemble des objets créés pendant une analyse, afin qu'elle soit reproductible, que les coûts de création puisse être évalués à l'avance et évalués en fonction de l'indispensabilité de chaque objet et enfin que les étapes nécessitant une validation (une catégorisation, mais aussi dans certains cas, une segmentation, des regroupements, etc.) puissent être soulignés ;
- d'installer un dialogue entre différentes pratiques analytiques ;
- aux créateurs d'outils d'analyse de bénéficier d'un cadre de référence illustrant les besoins analytiques et de leur fournir une proposition de modèle permettant d'y répondre ;
- d'identifier plus précisément les défis d'implémentation que nous devons prendre en compte afin de permettre une analyse instrumentée par un artefact informatique qui ne se dresse pas en obstacle au travail du chercheur.

Dans cet article nous examinons d'abord les spécificités d'analyse du domaine du CSCL, les outils d'analyse existants ainsi que les artefacts créés (que ce soit ou non au travers d'une instrumentation informatique) par différents chercheurs lors de l'analyse. Nous en dégageons la notion de *rejouable*, un objet qui permet de répondre de manière générique à un grand nombre de besoins d'analyse. Nous décrivons en détail les propriétés de cet objet ainsi que les opérations auxquelles il peut être soumis. Nous illustrons enfin son utilisation dans un cadre concret et examinerons ses limites et présentons les difficultés que nous avons éprouvées lors d'une première implémentation.

Problématique et état de l'art

Obtenir une vue d'ensemble de tous les artefacts qu'un chercheur pourrait vouloir créer est un défi considérable. Un tour d'horizon de différentes pratiques est facile à effectuer (Dyke *et al.*

2009b) mais ne garantit pas d'avoir une couverture complète. Dans cette section, nous examinerons d'une part la démarche analytique globale telle qu'elle est décrite par différents praticiens, et d'autre part les différentes formes de représentations créées lors de l'analyse ainsi que les outils qui permettent de les créer. Nous préciserons cette étude au contexte du CSCL, toujours en vu de répondre à la question « comment le chercheur met-il en œuvre cette pratique ? » En focalisant sur cette question, d'ordre très pragmatique nous éviterons de nous disperser dans les questions plus larges de « que doit faire le chercheur ? » et « pourquoi est-ce que le chercheur fait cela ? » qui sont respectivement d'ordre méthodologique et épistémologique. Ces questions vont au delà du cadre de cet article.

Qu'est-ce que l'analyse ?

Selon Thomas et Cook (Thomas et Cook, 2005), l'analyse est un processus itératif impliquant un cycle de *cristallisation de connaissances* (aussi appelé *construction de sens*) dont les étapes sont le rassemblement d'informations, la re-représentation de l'information sous une forme qui aide l'analyse, le développement d'une nouvelle compréhension (*insight*) au travers de la manipulation de cette représentation et la création d'un nouvel artefact (*knowledge object*) basé sur cette compréhension. Ils nomment *discours analytique* le dialogue médiatisé par la technologie entre un analyste et ses données. Ce discours analytique crée tour à tour des représentations, dont certaines seront des *produits* qui distillent les données et leur interprétation de manière à les communiquer à un public plus large.

Les coûts mis en œuvre lors d'opérations de construction de sens sont soulevés par Russel *et al.* (Russel *et al.*, 1993) afin d'évaluer la pertinence de l'assistance informatique aux différentes étapes de la démarche. Ils décrivent le *complexe du cycle d'apprentissage* (*learning loop complex*) dont les étapes sont la définition d'une représentation de données, l'encodage des données dans cette représentation, l'identification des *résidus* (les données auxquelles la représentation ne convient pas) et l'adaptation de la représentation.

On retrouve cette notion de construction de sens dans les travaux sur l'analyse qualitative de Tesch (Tesch, 1990). La démarche du chercheur est décrite comme consistant à décontextualiser des extraits de corpus et à les manipuler pour les comparer, les associer et les restructurer afin de leur donner du sens.

Dans le cadre des travaux sur l'analyse exploratoire de données séquentielles de Sanderson et Fisher (Sanderson et Fisher, 1994)(Sanderson et Fisher, 1996), une double boucle analytique

est mise en évidence. La première concerne la démarche scientifique qui consiste à poser des questions, récolter des données et les analyser, le tout dans un cadre épistémologique qui définit la nature des questions à poser, la manière de récolter les données ainsi que les types de résultats analytiques qui sont acceptables. La deuxième, intérieure à la première et spécifique à l'étape d'analyse, concerne la création de produits transformés à partir des données récoltées. Sanderson et Fisher définissent ces opérations de création comme étant les *8 Cs* : morceaux (*chunks*), commentaires, codes, connections, comparaisons, contraintes, conversions et calculs.

Les *morceaux* concernent la segmentation et la re-segmentation, parfois dans des structures hiérarchiques. Les *commentaires* sont des annotations informelles attachées à des données, à des morceaux ou à d'autres produits intermédiaires. Les *codes* sont des abstractions des données dans un vocabulaire contrôlé et ayant une définition précise. Les *connections* décrivent les relations entre les données, comme la temporalité, la structure implicite créée par l'artefact informatique qui a produit les données ou encore la synchronisation entre différents médias. Les *comparaisons* examinent l'effet de traitements différenciés sur les données, par exemple de comparer le même codage entre deux analystes ou de comparer deux séries de données issues de conditions expérimentales différentes ou encore d'examiner la différence entre un scénario prédictif et le scénario descriptif suivi (Choquet et Iksal, 2007). Les *contraintes* appliquées aux données permettent de les filtrer afin, par exemple, de focaliser de plus près sur un intervalle particulier, un médium particulier ou les données codées avec un certain mot clé. Les *conversions* transforment les données vers une autre forme, comme un changement d'unité d'analyse ou une représentation sous forme de graphe ou de ligne du temps. Enfin, les *calculs* réduisent les données à des représentations sommaires comme des décomptes ou des tests statistiques.

Sanderson et Fisher remarquent que d'éffectuer ces différentes opérations dans un ordre *ad hoc* (par opposition à dans un ordre prédéfini) permet de créer une approche méthodologique adaptée à une question analytique particulière et ajoutent que les analyses les plus riches combinent souvent toutes ces opérations (Sanderson et Fisher, 1996). Cette vision est en contraste avec le cycle d'analyse proposé par Harrer *et al.* (Harrer *et al.*, 2007) composé des opérations séquentielles de *capture*, *segmentation*, *annotation* (incluant aussi le codage), d'*analyse qualitative*, *quantitative* et *statistique*, de *visualisation* et d'*interprétation*.

Enfin, dans une perspective d'analyse en CSCL, les travaux de Suthers (Suthers, 2006)(Suthers *et al.*, 2007)(Suthers et Medina, 2008) proposent une représentation générique,

le *graphe de contingences*, qui peut être adapté à différentes épistémologies et méthodologies en CSCL en fonction de leurs spécificités individuelles et qui peut servir de *boundary object* entre ces épistémologies. Ce graphe des contingences décrit des nœuds qui représentent des *coordinations médiatisées* (des actions prises dans un environnement partagé) et des arcs entre ces nœuds qui sont des *contingences* qui décrivent en quoi une coordination médiatisée est objectivement contingente sur le contexte (d'autres coordinations médiatisées) dans lequel elle a été produite. A partir de ces contingences objectives, qui peuvent être des dépendances temporelles, des dépendances médiatiques inhérentes (par exemple la structure de réponse dans un forum) ou encore des relations d'ordre sémantique, le chercheur peut effectuer des interprétations de l'intersubjectivité (construction collaborative d'idées, transfert de connaissances, etc.) présente au cours d'une séance. Grâce à cette représentation, Suthers explique qu'il devient possible d'établir la trajectoire des acteurs, des idées qu'ils expriment ainsi que des artefacts qu'ils produisent. « C'est dans les confluences des trajectoires qu'on peut trouver les transformations qui peuvent être indicatrices d'apprentissages collaboratifs » (Suthers et Medina, 2008).

Dans ces travaux, plusieurs concepts clés se croisent. L'analyse consiste en la création itérative d'artefacts qui réifient la compréhension du chercheur ou lui permettent d'obtenir une nouvelle compréhension de son corpus. Passer d'un artefact à un autre requiert des transformations de natures diverses. Si les huit Cs paraissent bien recouvrir l'ensemble des manipulations qui peuvent s'effectuer sur ces artefacts, ils ne sont cependant pas directement utilisables en raison de la variété et du recoupement des opérations comprises dans chaque type de manipulation : par exemple, comme nous verrons en examinent les outils existants, les *connexions* peuvent être mises en évidence par des opérations de visualisation, d'explicitation de relations, de transformation ou encore de synchronisation ; et ces mêmes opérations peuvent aussi permettre les *comparaisons*, les *calculs*, et les *conversions*.

La spécificité de l'analyse dans le domaine du CSCL

Notre domaine d'application principal est celui du CSCL. Ce domaine se démarque par la présence de la médiatisation informatique, de la collaboration et de l'apprentissage. Nous désirons cependant décrire les caractéristiques d'un object analytique aussi générique que possible. Examinons chacun de ces aspects pour voir la spécificité de son apport dans

l'analyse et les possibilités de relaxation permettant d'appliquer l'approche à d'autres domaines proches comme les EIAH, le CSCW (travail collaboratif assisté par ordinateur) ou l'apprentissage collaboratif non-médiatisé.

La médiatisation informatique produit des traces particulières par rapport aux autres enregistrements (Laflaquière et Prié, 2009). La totalité de l'activité n'est pas enregistrée et ce qui est enregistré l'est avec une intention de collecte de la part du concepteur du système informatique utilisé. Elle est donc directement porteuse du sens donné par l'effet de l'action tracée sur le système (pour autant que cet effet sur le système soit documenté). Cette trace est d'autant plus riche que la médiatisation informatique « transforme la communication en substance » (Dillenbourg, 2005), rendant explicites et directement manipulables par des moyens informatiques les interactions langagières. Dans le cas d'autres formes de médiatisations (comme des outils graphiques ou de gestes avec la souris), ou lors d'interactions enregistrées au travers de plusieurs médias différents, cette richesse peut cependant se révéler difficile à replacer en contexte et donc à comprendre pour le chercheur (Dyke *et al.*, 2007). Une fois l'outil appareillé, les enregistrements numériques requièrent très peu d'actions supplémentaires et permettent la création de corpus sur une période longue intégrant aussi bien des données synchrones qu'asynchrones. De nombreuses situations mélangent interactions médiatisées par ordinateur et interactions en face-à-face (LEAD, 2006). Si la trace numérique peut présenter des avantages pour l'analyse, nous ne pouvons compter sur sa présence exclusive et devons donc tenir compte d'autres modalités d'enregistrement comme l'audio et la vidéo.

La collaboration est intéressante par les interactions humaines qu'elle engendre. Ces interactions, plus encore que d'autres actions, sont contingentes à un contexte compliqué, rendant nécessaire leur re-contextualisation pour être compréhensibles par un analyste. De plus, la médiatisation informatique créant des communications consultables de manière persistante, le contexte ne se situe pas forcément que dans la temporalité, exigeant parfois une analyse de l'interaction allant au delà des unités de tours de parole et de paires d'adjacence (Suthers, 2007).

Enfin, la dimension de l'apprentissage rend l'analyse particulière. D'une part, le contexte devient encore plus indispensable pour l'analyse car l'enregistrement est situé dans un contexte d'apprentissage qu'il convient de définir et de restituer afin que l'analyste puisse l'exploiter (Reffay, 2007). D'autre part, comme l'explique Suthers (Suthers, 2006), l'intérêt pour la médiation et la collaboration en rapport avec l'apprentissage dépend de

l'épistémologie dans laquelle se place le chercheur. En particulier, « *The focus [of CSCL] is defined by what aspect of human collaborative activity we examine and try to make sense of: intersubjective meaning making* » (Suthers, 2006, p. 322). Que nous acceptons ou non la construction de sens intersubjective comme l'aspect approprié à étudier, il n'en demeure pas moins que c'est au chercheur effectuant l'analyse de définir son objet d'étude en fonction de ses croyances épistémologiques.

Ce point est très important. En effet, les outils d'analyse « influencent la logique du chercheur » (Savoie-Zajc, 2000), (cité par (Teutsch *et al.*, 2008)). Si nous souhaitons définir un objet générique, il nous incombe de minimiser les décisions que nous prenons à la place du chercheur. Ainsi nous ne nous prononçons pas sur les cadres épistémologiques et méthodologiques, laissant au chercheur le soin de valider son approche méthodologique et de choisir les aspects de l'interaction humaine qui l'intéressent.

Nous pouvons donc chercher un objet qui permette de servir l'analyse de *l'interaction humaine située*, considérant les domaines de l'interaction entre humains (par opposition à un environnement informatique), de l'action médiatisée et de l'action en contexte d'apprentissage comme des cas présentant des difficultés supplémentaires en termes de quantité de données, de complexité de l'interaction et de sa situation, qui accroissent la difficulté de l'analyse sans en changer sa nature fondamentale. Nous focaliserons néanmoins sur les problématiques du CSCL d'une part car cette discipline cumule les difficultés présentées et d'autre part car c'est celle que nous connaissons le mieux.

Les outils et artefacts d'analyse existants

Lors de l'analyse de l'interaction humaine située, le chercheur crée des artefacts de natures variées, parfois avec une assistance informatique. Dans cette section, nous effectuons un état de l'art des outils d'analyse existants, ainsi que des différents aspects des artefacts qui peuvent être créés, que ce soit avec ces outils ou avec des méthodes plus *ad hoc*, afin de dépeindre dans les grandes lignes les caractéristiques souhaitées de notre objet générique d'étude qui décrit ces artefacts ainsi que les façons de le créer et de le transformer.

Etat de l'art

Les outils assistant l'analyse des interactions peuvent être classés par la nature des données qu'ils sont destinés à traiter. On peut distinguer les enregistrements audio-vidéo (combinant

les registres langagiers et gestuels), les traces numériques d'outils informatiques (principalement issus des domaines de l'interaction homme-machine, de la collaboration et de l'apprentissage), ainsi que des données d'ordre « métrique » comme la fréquence cardiaque, l'activité cérébrale, l'oculométrie, et la position dans l'espace (issus de la psychologie expérimentale, de la psychologie cognitive et de l'informatique mobile).

Les outils d'annotation et de transcription de la vidéo sont nombreux et un état de l'art considérable a été fait par Dybkjaer *et al.* (Dybkjaer *et al.*, 2001) dans le cadre d'un groupe de travail sur l'interactivité multimodale. Dans des travaux subséquents Reidsma et ses collègues (Reidsma *et al.*, 2005) décrivent les besoins spécifiques liés à l'annotation multimodale. Parmi les outils d'annotation vidéo, on peut citer Elan (Wittenburg *et al.*, 2003) , Anvil (Kipp, 1996) et NXT (Carletta *et al.* 2003). De manière générale, ces outils permettent d'ajouter des annotations à des intervalles sur une ligne de temps. Ils diffèrent par leur interface graphique, par la nature des intervalles (contigus, avec chevauchements, sur des lignes de temps parallèles, construits de manière hiérarchique) ainsi que par la nature des annotations (libres, symboliques, langagières, prises parmi une liste finie, structurées de manière hiérarchique). Viennent ensuite des fonctionnalités complémentaires d'exploitation, comme la recherche, la recherche de séquences, le calcul de statistique et l'export vers Excel, SPSS, etc.

Les outils permettant de traiter les traces numériques ont tendance à incorporer une composante de calcul et/ou une composante de visualisation. Le *Synchronized Analysis Workshop* (ou SAW) (Goodman *et al.*, 2006) est un outil de visualisation et d'analyse pour permettre la navigation synchronisées de sources de données multiples, fournissant entre autres, une visualisation symbolique graphique sur une ligne de temps des évènements de la trace, et des possibilités de filtrage et de partage avec d'autre chercheurs. CORDTRA (« *Chronological representations of discourse and tool-related activity* ») (Hmelo-Silver, 2003) est un script intégré à Excel qui permet de créer des visualisations sous forme de lignes de temps parallèles. Travis (May *et al.*, 2008) produit des lignes de temps permettant de voir les durées de rédaction et de lecture de messages dans des forums. ViCoDiLi (Teutsch *et al.*, 2008) est un outil d'exploration et de visualisation de corpus d'échanges en ligne qui permet de recontextualiser des échanges dans la structure d'interventions de l'outil mais aussi de les décontextualiser afin de les rassembler dans une structure ayant un sens analytique. Cet outil dispose en outre de fonctions de recherche et d'anonymisation. Le Session Browser du projet LISTEN (Mostow *et al.*, 2005) permet de rechercher des évènements dans une trace puis les

recontextualiser en reproduisant la *chaine des évènements ancêtres*, l'événement A étant l'ancêtre de B si B démarre pendant A.

La combinaison de vidéo et traces numériques comme source de données mène à de nouveaux outils. I-Observe (Badre *et al.*, 1995) est un outil pour l'analyse d'interactions homme-machine dans lequel la trace numérique peut être filtrée et visualisée afin de servir d'index pour explorer la vidéo. ActivityLens (Fiotakis *et al.*, 2007) est un outil destiné à analyser des traces numériques en parallèle avec des enregistrements vidéo dans un cadre de théorie de l'activité. Il permet l'ajout d'annotations ainsi que la reconstitution de l'activité sur trois niveaux, regroupant les opérations en actions et les actions en activités.

Deux outils aux fonctionnalités proches assistent principalement le rejouage et la synchronisation de données issues de sources multiples. Replayer (Morrison *et al.*, 2006) est un outil conçu pour l'évaluation d'applications mobiles et permet la synchronisation de visualisations d'une variété de sources de données telles que vidéos, traces numériques, données géographiques et autres données métriques. DRS (Greenhalgh *et al.*, 2007) se positionne dans le même domaine et intègre en outre les fonctionnalités usuelles des outils d'annotation vidéo.

ABSTRACT (Analysis of Behavior and Situation for menTal Representation Assessment and Cognitive acTivity modelling)(Georgeon *et al.*, 2006) est un système conçu pour faciliter la découverte de connaissances à partir de traces d'activité. Il intègre une visualisation symbolique en ligne de temps avec un moteur d'inférence capable de générer des éléments de plus haut niveau à partir de règles. Il permet aussi la synchronisation avec de la vidéo.

Enfin, nous pouvons examiner quelques outils de traitement de la trace qui ne font pas partie des familles citées plus haut. Le rejoueur de DREW (Corbel *et al.*, 2003) effectue une relecture de la trace produite par le logiciel DREW (un outil de communication médiatisée synchrone intégrant des modules tels que *chat*, *éditeur de texte partagé* et *graphes d'argumentation*) et l'utilise pour reconstituer à l'écran ce que les participants pouvaient voir. Cette reconstitution est similaire au résultat d'une capture d'écran, mais prend largement moins d'espace mémoire qu'une vidéo, permet d'explorer tout l'environnement à un instant donné (par exemple en déplaçant l'ascenseur ou en ré-agençant les fenêtres) et peut avoir une interface graphique légèrement différente de l'originale (permettant par exemple d'attribuer des couleurs différentes aux interventions des différents intervenants afin de faciliter l'analyse). Le *Knowledge Space Visualiser* (KSV) (Teplovs et Scardamalia, 2007) est un outil qui place les différents objets qui constituent un forum sur un graphe ou les arcs sont des

relations entre ces objets, comme la proximité sémantique et la relation dans la structure de réponse. Ce graphe est ensuite projeté en deux dimensions. Il permet aussi la recherche par mot clé et la sélection d'un nœud permet d'obtenir des informations supplémentaires (dans le cas d'un message, auteur, contenu, date posté, etc.). D'autres structures de graphe se retrouvent dans l'analyse de réseaux sociaux (Harrer *et al.*, 2006) ou la description des chemins parcourus le plus fréquemment sur un site web (Romero *et al.*, 2008). La catégorisation dans le cadre de l'analyse de contenu (de Wever *et al.*, 2006) étant très fréquemment utilisée, il existe des outils effectuant une catégorisation automatisée basée sur des algorithmes d'apprentissage machine et de traitement du langage, par exemple TagHelper Tool (Rosé *et al.*, 2008).

Les différents aspects des artefacts d'analyse

Afin de faire la synthèse de ces différents outils et pour présenter quelques autres artefacts créés sans support informatique dédié nous avons dégagé les thématiques qui semblent intéressantes : la contextualisation, la visualisation, la transformation, l'enrichissement, la comparaison, et les aggregations. Les fonctionnalités des outils présentés ci-dessus vis-à-vis de certaines de ces thématiques sont résumées dans le Tableau 1. A titre comparatif, nous faisons aussi figurer le logiciel Excel, les lecteurs audio et vidéo ainsi que les graphes de contingences tels que présentés par Suthers. Remarquons qu'aucun de ces outils n'adresse l'ensemble des fonctionnalités, principalement du fait des spécificités différentes des outils à des méthodologies d'analyse précises.

	Contextualisation	Visualisation	Transformation	Enrichissement
Elan	Rejouage, Synchronisation	Ligne de temps, Tableau	Création d'intervalles	Annotations, Catégories
Anvil	Rejouage, Synchronisation	Ligne de temps	Création d'intervalles	Annotations, Catégories, Collections, Structures
NXT	Rejouage, Synchronisation	Ligne de temps, Texte	Création d'intervalles et de zones de sélection, Filtrage, Recherche de séquences	Annotations, Catégories, Collections, Structures
SAW	Synchronisation	Ligne de temps, Tableau	Filtrage, Insertion d'évènements	-
CORDTRA	-	Ligne de temps	-	-
Travis	-	Ligne de temps	Filtrage	-
ViCoDiLi	Structure	Arbre	Filtrage	Collections
LISTEN session browser	Structure	Arbre	Filtrage	-

I-Observe	Rejouage, Synchronisation	Ligne de temps	Filtrage, Recherche de séquences	-
ActivityLens	Rejouage, Synchronisation	Tableau	Filtrage, Regroupement d'évènements	Annotations
Replayer	Rejouage, Synchronisation	Ligne de temps, Tableau, Courbe de temps, Carte	-	-
DRS	Rejouage, Synchronisation	Ligne de temps, Tableau, Courbe de temps	Création d'intervalles	Annotations, Catégories
ABSTRACT	Rejouage, Synchronisation	Ligne de temps	Requêtes de filtrage et de construction	-
Rejoueur DREW	Rejouage	Image 2d	-	-
KSV	Structure	Graphe	Filtrage	-
Réseaux Sociaux	Non	Graphe	-	-
Graphes de chemins	Nœuds ordonnés	Graphe	-	-
TagHelper Tool	-	Tableau	-	Catégorisation automatique
Excel	-	Tableau	Insertion de lignes	Ajout de colonnes
Lecteur Video	Rejouage	Image 2d	-	-
Lecteur Audio	Rejouage, Synchronisation	Courbe de temps	-	-
Graphe de contingences	-	Ligne de temps	-	Structure

Tableau 1 • Comparaison des fonctionalités de divers outils d'analyse en termes de contextualisation, visualisation, transformation et enrichissement.

La contextualisation

L'une des nécessités analytiques fortes que nous avons évoquées, liée à la nature contingente des données, est le besoin de contextualisation ou de recontextualisation, afin de présenter ces données au chercheur sous une forme qui soit compréhensible et qui permette de comprendre une interaction sur le plan pragmatique. Cette contextualisation rentre dans l'aspect *connections* des huit Cs. En premier lieu, ces données sont situées dans le temps. Presque tous les outils que nous avons observés cherchent à produire des artefacts qui repositionnent ces données dans une temporalité, que ce soit par des visualisations présentant le temps en abscisse ou en ordonnée, ou, pour les outils qui permettent de rejouer de la vidéo ainsi que pour le rejoueur de DREW, en permettant de « rejouer » via une télécommande présentant les fonctions *play* et *pause*.

Lorsqu'un outil permet la vision simultanée de plusieurs artefacts qui représentent, par exemple, les mêmes données sous différentes formes ou qui représentent des données co-occurrentes issues de médias ou de modes différents, il offre souvent des possibilités de synchronisation temporelle (cf. Figure 30). Cette synchronisation permet de combiner des vues multiples afin de mieux comprendre une situation. D'autre part, certains artefacts sont susceptibles de servir d'index pour d'autres (Suthers *et al.*, 2007) ; par exemple une transcription peut assister la navigation dans une vidéo. Enfin, la nature itérative de l'analyse conduit à la construction d'artefacts de plus en plus abstraits. La synchronisation permet de maintenir le lien entre les différents niveaux d'abstraction et avec les données primaires afin de confirmer la véracité de connaissances cristallisées à partir d'artefacts secondaires.

Figure 30 • Cet exemple du logiciel Replayer présente cinq artefacts synchronisés : une ligne de temps présentant des éléments sous forme symbolique, une vue aérienne avec certaines positions mises en évidence, deux vidéos et une ligne de temps plus simple permettant de sélectionner des intervalles. Les mêmes évènements sont mis en évidence dans tous les artefacts hormis les vidéo.

Une autre forme de contextualisation se révèle dans la structure inhérente aux médias utilisés. Si une conversation verbale suit principalement un ordre chronologique où la proximité temporelle est un élément contextuel très fort, ce n'est pas forcément le cas lors de communications médiatisés par des forums ou des graphes d'argumentation. De même, des évènements peuvent être situés dans une hiérarchie d'évènements, comme c'est le cas pour les données pour lesquelles est conçu le Session Browser de LISTEN. Il est donc utile que les logiciels d'analyse puissent comprendre et rétablir cette structure, comme le fait par exemple l'outil ViCoDiLi qui permet de rétablir n'importe quelle structure de forum.

Enfin, dans un cadre plus large, il ne faut pas oublier que le contexte dans lequel un recueil de données s'est effectué est souvent indispensable à l'analyse (Reffay, 2007). Il s'agit alors de pouvoir se référer à la situation pédagogique, aux entretiens avec les sujets observés, au niveau de connaissance et de familiarité d'un groupe, etc.

La visualisation

Voyons maintenant les différentes formes de visualisation que ces artefacts peuvent utiliser. On recense :

- des images en deux dimensions (qui vont changer avec le temps) comme pour les lecteurs vidéo, le rejoueur de DREW ou encore les vues aériennes de Replayer ;
- des tableaux contenant des lignes et des colonnes, à la manière d'Excel ;
- des lignes de temps présentant chaque évènements sous forme symbolique (SAW, CORDTRA, ABSTRACT, Travis) ;
- des lignes de temps permettant de sélectionner des intervalles et d'y ajouter du texte (Elan, Anvil, DRS, NXT)
- des courbes présentant une métrique variant au cours du temps (DRS, Replayer et les lecteurs audio qui proposent une vue de l'intensité du signal sonore au cours du temps) ;
- du texte sur lequel il est possible de poser des ancres (NXT) ;
- des arbres (des tableaux dont les lignes sont aussi des nœuds d'arbre et dont les fils sont indentés par rapport à leurs parents) (ViCoDiLi et le Session Browser);
- des réseaux présentant les liens entre certains objets (comme des évènements, des documents ou des individus) (Analyse de Réseaux Sociaux, KSV).

La majorité de ces visualisations contribuent à la contextualisation, que ce soit par la représentation de la structure ou celle du temps. La représentation du temps peut prendre plusieurs formes. Certains artefacts ne présentent qu'un « arrêt sur image », ne permettant d'appréhender le contexte temporel qu'à l'aide de rejouage. Le lecteur audio est un cas extrême : lorsqu'il est sur pause, rien n'est présenté à l'auditeur, ce n'est que lors du rejouage qu'une représentation sonore est produite. Souvent, les évènements séquentiels sont représentés de manière séquentielle, par des lignes consécutives dans un tableau ou par des éléments symboliques adjacents dans une ligne de temps. Enfin, lorsqu'une visualisation présente des évènements mais ne représente pas la proximité temporelle par une proximité spatiale, la numérotation de ces évènements contribue à une notion de séquentialité (Blake et Rapanotti, 2001). Dans tous les cas ou les évènements sont représentés dans la visualisation, ils peuvent servir d'éléments de navigation dans la synchronisation entre artefacts selon un principe de « *brush and link* » (Becker et Cleveland, 1987) : la sélection d'un objet dans une vue met en évidence cet objet dans toutes les autres vues.

L'utilité de ces différentes formes de visualisation est aussi de présenter les données au chercheur de façon à rendre l'aspect de l'interaction qui l'intéresse facile à voir : « *The simple act of placing information on a timeline or a map can generate clarity and profound insight.* » (Thomas et Cook, 2005, p. 37). Les visualisations appropriées présentent les avantages suivants (Card *et al.*, 1999) : l'augmentation des ressources cognitives, une réduction du temps de recherche, l'augmentation de la reconnaissance de régularités, le soutien a une inférence perceptuelle de relations, la possibilité d'avoir une vue d'ensemble d'un grand nombre d'évènements et de fournir un artefact manipulable qui, au contraire d'un diagramme statique permet l'exploration d'un espace de valeurs de paramètres.

La transformation

L'acte de transformation prend un artefact composé d'un certain nombre d'éléments (par exemple des évènements) et produit un nouvel artefact composé d'éléments supplémentaires, d'un nombre restreint d'éléments ou d'éléments à une granularité différente. Cela recouvre plusieurs aspects des huit Cs : *morceaux*, *contraintes*, *connections* et *conversions*. Ces transformations peuvent être manuelles, assistées par un outil ou automatisées selon la disponibilité d'un outil capable d'assister la transformation et la capacité de l'état de l'art à effectuer cette transformation sans intervention humaine.

Dans la création de morceaux, on trouve notamment tous les outils qui permettent de sélectionner des intervalles sur une ligne de temps (Elan, Anvil, DRS) ou de sélectionner et subsumer une série d'évènements (ActivityLens), ou encore d'insérer de nouveaux éléments (Excel, SAW). Les capacités de conversion sont très proches et souvent automatisées comme la recherche de séquences – et la création subséquente de nouveaux éléments représentant ces séquences – trouvée dans les outils permettant de faire des requêtes complexes tels que ABSTRACT (cf. Figure 31), I-Observe et NXT. Des requêtes plus simples permettent d'imposer des contraintes aux données comme le filtrage ou la sélection d'un intervalle limité (*slicing*, respectivement *segmentation* (Mostow et Beck, 2006)) ou encore de calculer des propriétés supplémentaires d'un élément (par exemple un texte qui décrive mieux à un humain l'effet d'une action). Enfin, des *connexions* sont créés dans les transformations de fusion qui combinent des données issues de sources différentes dans un seul artefact (par exemple dans SAW) que ce soit en triant par le temps ou en exhibant d'autres connexions (appartenance à un même arbre de discussion par exemple).

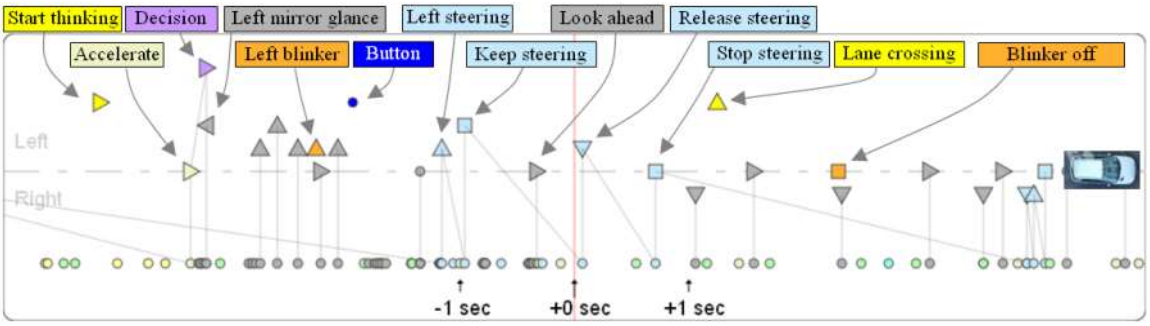


Figure 31 • Cet exemple du logiciel ABSTRACT présente la trace primaire (en bas) et des éléments de plus haut niveaux calculés à partir de cette trace primaire (en haut.)

Notons que, parmi les *conversions*, celles qui consistent en une simple visualisation du même artefact d'une façon différente ne correspondent pas dans notre description à une transformation (car elles ne modifient pas les éléments composant l'artefact).

L'enrichissement

Dans le cycle de cristallisation de connaissances, la génération de nouvelles connaissances est suivie de leur inscription dans de nouveaux artefacts. Hormis la création de nouveaux éléments, cette inscription peut aussi se faire directement « sur » les éléments existant déjà. Ces enrichissements peuvent prendre la forme de *commentaires*, de *codes* et de *connexions*. Dans les outils d'annotation vidéo notamment, les enrichissements s'ajoutent directement sur des intervalles de temps. Dans d'autres outils, cela se fait sur des évènements (ou sur des lignes de données dans un tableau).

Les commentaires, ou annotations permettent au chercheur de réifier certaines connaissances, afin de servir d'aide mémoire pour la suite ou de les partager avec d'autres chercheurs. Elles peuvent aussi anticiper un besoin subséquent d'imposer à ces annotations un vocabulaire contrôlé en vue de passer à une forme de catégorisation (par exemple lorsqu'il s'avère que les mêmes annotations sont fréquemment utilisées et qu'il s'agit en fait de catégories).

Plus encore qu'un vocabulaire contrôlé, certains outils (par exemple, NXT) permettent de créer de véritables ontologies de catégories afin que l'attribution d'une catégorie à tel élément (par exemple de dire que tel mot est un adjectif) puisse permettre des inférences subséquentes (que cet adjectif peut être combiné avec d'autres mots pour former un groupe nominal). Dans l'examen de l'état de l'art des outils d'annotation vidéo, les structures qu'il est possible d'imposer aux annotations, codes et catégories sont souvent critiquées. En effet, d'une part elles imposent à l'utilisateur du logiciel un modèle mental complexe qu'il devra s'approprier, d'autre part, elles peuvent contraindre l'utilisateur à une certaine façon de fonctionner (par

exemple établir à priori une liste de catégories qu'il ne connaît pas encore). Enfin, l'absence de telles structures ne correspond souvent pas à une liberté de création de structure implicite par le chercheur mais à une structure simplifiée explicite (par exemple qui implique que l'on ne peut attribuer deux catégories différentes à un même évènement) qui est toute aussi contraignante qu'une structure explicite complexe. On retrouve ces notions de codage dans l'analyse de traces numériques (de *chat* notamment). L'ubiquité de cette technique a mené à plusieurs outils visant à automatiser ce codage grâce à l'apprentissage machine tels que TagHelper.

Une fonctionnalité voisine consiste en la création de collections d'éléments de natures similaires. Dans ViCoDiLi, cela se fait en rajoutant des éléments à un « panier ». Dans les outils d'annotation vidéo, particulièrement dans le cadre de l'analyse conversationnelle, cela se fait en rajoutant un mot clé à un élément qui fonctionne comme un *tag*, c'est à dire une forme de rajout de mot clé qui permet de faciliter le regroupement en catégories et l'indexation d'un corpus. La collection résultante peut être vue comme la recherche de tous les éléments ayant un tag donné.

Un type d'enrichissement particulier est la mise en évidence manuelle ou automatisée de liens entre éléments dans l'optique d'une visualisation de ces liens par un graphe et d'une analyse éventuelle de ce graphe en utilisant les liens pour suivre la trace d'un acteur, d'une idée ou d'un artefact de médiation (Suthers et Medina, 2008). On retrouve cette notion de lien dans l'outil de visualisation KSV ainsi que dans les travaux de chercheurs voulant mettre en évidence les contingences entre différents évènements (Suthers *et al.*, 2007)(Lund *et al.*, 2008)(cf. Figure 32).

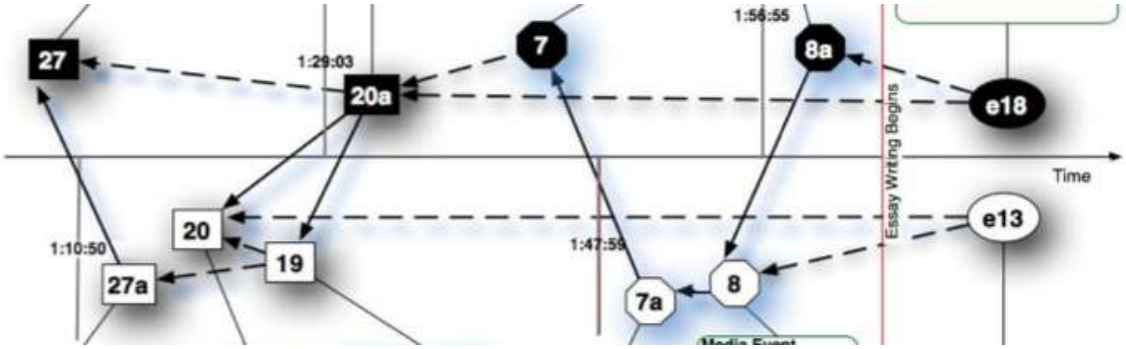


Figure 32 • Graphe de contingences illustrant les dépendances entre les contributions de deux étudiants (en haut et en bas respectivement), afin de tracer le transfert de connaissances de l'un à l'autre (Suthers *et al.*, 2007)

La comparaison

Des huit Cs de Sanderson et Fisher, deux n'ont pas encore été abordés et ne figurent pas dans le Tableau 1 de comparaison entre différents outils. Le premier est la *comparaison*. Ces comparaisons peuvent prendre plusieurs formes. Une analyse étant la rencontre entre un chercheur, un corpus et une méthodologie (Dyke *et al*, 2007) et un corpus de CSCL mettant typiquement en scène un ensemble de groupes d'apprenants, d'outils et de scénarios pédagogique, la variation de seulement l'un de ces paramètres peut éclairer son rôle ou servir à confirmer des résultats. Le codage par deux chercheurs peut permettre de répartir la charge de travail ou de vérifier ce codage par le méthode des juges (de Wever *et al.*, 2006). L'application parallèle ou séquentielle de méthodologies d'analyse différentes permet d'obtenir de nouveaux résultats ou de confirmer des résultats par triangulation (Suthers, 2006). Enfin, la comparaison de groupes, d'outils et de scénarios peut permettre d'obtenir des informations statistiques ou d'utiliser des études de cas pour contraster qualitativement deux situations.

Malgré la nécessité évidente de la comparaison d'artefacts d'analyse divers, peu d'outils soutiennent explicitement cette opération. Seul SAW prend en compte la nécessité de travail collaboratif entre chercheurs, leur permettant de partager des artefacts d'analyse et des résultats partiels (tous les logiciels qui permettent d'enregistrer un fichier informatique permettent aussi le partage de ce fichier, mais nous n'avons pas vu d'intention explicite de collaboration). Au mieux, pour comparer deux artefacts, peut-on les ouvrir côte-à-côte et les comparer ainsi, ou utiliser des outils statistiques pour établir des résultats sur des groupes multiples. Dans l'assistance à la tâche de comparaison d'enrichissements, on peut noter l'outil Excel, grâce auquel il est possible de copier le codage d'un chercheur et le coller à côté de celui d'un autre.

L'agrégation

Le dernier des huit Cs à examiner est l'aspect des *calculs*. Ces calculs sont présents dans la plupart des logiciels sous forme d'agrégations et de statistiques plus ou moins complexes. Leur nature est typiquement de prendre un artefact ou une partie d'artefact et de le *résumer* à des valeurs numériques qui s'appliquent à son ensemble (par exemple nombre d'interventions, nombre d'interventions par apprenant, taux de participation, taux de collaboration, etc.). On retrouve notamment ce genre d'artefact dans la littérature sur les indicateurs (Martinez *et al*., 2003). Nous ne nous attarderons peu sur ce concept d'agrégation

car son résultat est fondamentalement différent des autres artefacts que nous avons pu considérer. Il n'est plus temporellement situé, perdant ainsi quasi toute notion de contingence. Hormis la question de l'opération de calcul lui même, son traitement subséquent peut se faire à l'aide d'outils existants de statistiques et de visualisation ce qui est hors du cadre de cet article.

Nous pouvons cependant examiner le lien de parenté avec les données que nous avons nommées « métriques ». Ces données présentent la variation d'une ou plusieurs valeurs au cours du temps. L'analyse et le traitement de signaux (sonores notamment) passe souvent par la notion d'une fenêtre glissante : Une fenêtre d'un certain intervalle (par exemple une seconde) est glissée le long du signal par incréments successifs (par exemple un dixième de seconde) ; pour chaque de fenêtre, un calcul est effectué ; le résultat est une métrique prenant une valeur pour chaque fenêtre, cette valeur agrégeant des informations dans un voisinage de la taille de la fenêtre (une valeur tous les dixièmes de seconde calculé à partir d'un voisinage d'une seconde dans notre exemple).

Ce système de fenêtre glissante pourrait être adapté pour considérer la variation du résultat d'un calcul au fil du temps et montre qu'on est, dès lors. proche de la notion de transformation automatisée, déjà présentée.

La notion de rejouable

Dans la section précédente, nous avons présenté l'analyse comme étant un processus itératif de création d'artefacts. Nous avons examiné en détail les artefacts crées par les chercheurs à partir de traces d'interactions et avons dégagé les besoins de contextualisation, de transformation et création, de visualisation, d'enrichissement et de calcul sur ces artefacts. Dans cette section, nous présenterons un artefact générique répondant à ces besoins que nous nommons *rejouable*. Nous le spécifierons à un niveau qui permette d'informer l'implémentation sans pour autant la déterminer. Nous pensons qu'un objet défini à ce niveau d'abstraction peut aussi servir de modèle de réflexion par les chercheurs sur leur analyse afin de prédéfinir à l'avance les artefacts qu'ils ont besoin de créer, de déterminer la meilleure façon de construire ces artefacts et de communiquer et discuter de ces artefacts avec d'autres chercheurs.

Observable et trace

Afin de définir un artefact permettant l'analyse de traces d'interaction il convient de présenter ce que nous entendons par une *trace*. Dans une situation idéale, le chercheur observerait ce qui se passe et analyserait directement à partir de cette observation, ayant tout loisir d'examiner tout ce qui est *observable*. La nécessité pragmatique d'une période d'analyse plus longue que la mise en situation observée ainsi que la possibilité de capturer une grande quantité de données rend impossible l'étude directe de ce qui est observable. Il faut donc passer par l'enregistrement de ce qui était observable. Le résultat de cette enregistrement est une *trace* qui peut être issue d'un appareillage de collecte de traces sur un média informatique, d'appareils de mesure ou d'enregistrements audio et vidéo.

La difficulté d'analyse est alors dupliquée : il faut d'une part, à partir de cette trace, restituer autant que possible au chercheur ce qui était observable et d'autre part permettre au chercheur de construire de nouveaux artefacts dans le cycle de construction de sens qui constituera son analyse.

Rejouable

La majeure partie des artefacts que nous avons vus à la section précédente peuvent être rejoués et voir ce rejouage synchronisé avec celui d'autres artefacts. Ce sont donc, dans le cadre que nous proposons, des *rejouables*, c'est à dire des artefacts analytiques représentant des données issues de traces et qui peuvent être rejoués et synchronisés entre eux. Hormis les agrégations, les artefacts présentés qui ne bénéficient pas de rejouage synchronisé (par exemple les graphes de contingence) sont tout de même des rejouables en vue du fait qu'ils sont limités plus par leur implémentation que par leur nature même (ils sont présentés de telle manière qu'il serait possible de les rejouer et synchroniser à la main, sinon avec un assistance informatique). Comme nous avons vu, la synchronisation et le rejouage ne sont pas les seules caractéristiques des ces objets, mais la nature contingente et temporelle des données contenues dans la trace en fait des caractéristiques clés pour que ces objets soient utiles pour l'analyse du chercheur.

Il convient de distinguer deux choses : les données abstraites représentées dans un rejouable et leur représentations graphiques (ou parfois audio). Dans cet article, nous parlerons donc de *rejouable* et de *visualisation de rejouable* respectivement (cf. Figure 33). Nous pouvons donc examiner en détail la création et la transformation de rejouables avant de voir les conséquences pour la visualisation.

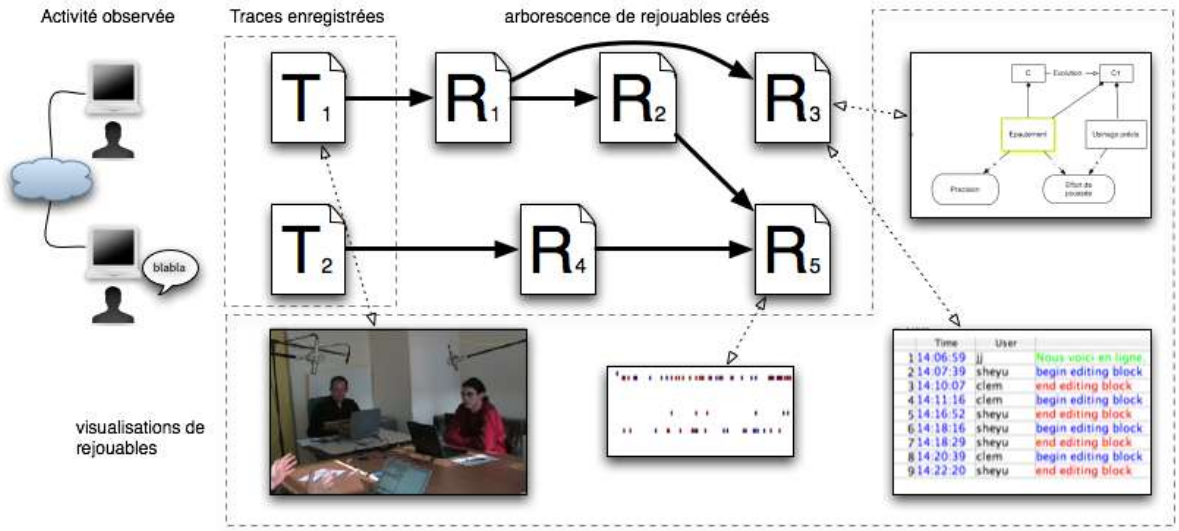


Figure 33 • Illustration d'une série de rejouables crées à partir de la trace enregistrée d'une activité observée. Chaque rejouable peut être visualisé d'une ou plusieurs façons.

Une modélisation de rejouable

La création de rejouables peut passer par la transformation automatique, la segmentation manuelle, la sélection ou le re-ordonnancement des éléments composant une trace ou un rejouable déjà existant. Cependant, comme presque tous les outils vu à la section précédente le montrent, il n'est pas toujours obligatoire d'effectuer une opération de création de rejouable avant de construire une visualisation servant l'analyse à partir d'une trace. Il en découle que la trace doit déjà être un rejouable.

On peut donc se poser trois questions : Est-ce que tous les rejouables sont des traces ? Peut-on utiliser bâtir notre modélisation de rejouable sur une modélisation de trace ? Quelles sont les caractéristiques qui vont différencier les rejouables des traces ?

A cette première question nous répondons « non ». En effet, si la trace est l'enregistrement de l'activité observable, certaines transformations (notamment la segmentation manuelle) créent des objets qui représentent l'interprétation du chercheur de ces données et non les données elles-mêmes. D'autre part, les enrichissements contiennent des informations tout aussi subjectives qui vont au delà du simple enregistrement. Une typologie de traces existante présente la notion de trace subjective (Choquet et Iksal, 2007), mais ne permet pas de représenter l'enrichissement subjectif de traces déjà existantes.

Les travaux de Settouti *et al.* (Settouti *et al.*, 2006) présentent un modèle mathématique de la trace qui, en vastement simplifié, définit la trace comme un ensemble d'évènements temporellement situés et de relations entre ces évènements ; ces évènements et relations sont

munis d'un modèle qui décrit les types d'évènements qui peuvent être trouvés dans la trace. Le niveau de formalisme de ce modèle est établi tel qu'il soit possible de définir un langage de transformation et de requête sur la trace. Nous souhaitons nous placer à un niveau plus haut ou le modèle de trace ne sert pas à assurer la faisabilité de calculs, mais à expliciter ou remplir les besoins de chercheurs. Pour ce faire, nous nous baserons directement sur le modèle de Settouti *et al.* qui nous fournit presque le format de donnée abstrait dont nous avons besoin. Un rejouable devient alors un ensemble d'évènements temporellement situés (suffisant pour le rejouage et la synchronisation) et de liens entre ces évènements (suffisant pour l'aspect *connexions* des huit Cs). Les évènements composant ce rejouable sont décrits par un ensemble de facettes nommées et typées. Les transformations consistent alors en la création de rejouables composés d'autres évènements (ou d'un sous-ensemble ou sur-ensemble des évènements d'un autre rejouable).

Un premier aspect qui n'est pas pris en compte par ce modèle est la particularité des enrichissements. Les enrichissements pourraient se faire par l'ajout de nouvelles facettes aux évènements dans un rejouable existant. Mais il serait alors difficile de partager les enrichissements entre chercheurs sans avoir à re-communiquer des données déjà partagées au préalable. Par ailleurs, supposons que dans Figure 33 une catégorie soit ajoutée à un événement du rejouable *R3* et que cet événement soit aussi présent dans les rejouables *T1, R1*, *R2* et *R5*. Il en découle que cette catégorie puisse aussi être ajoutée à cet événement dans les autres rejouables. Pour répondre à ce besoin, nous proposons : d'une part que chaque événement de rejouable soit considéré comme le représentant d'un événement observable enregistré dans une trace ; et d'autre part que les enrichissements soient enregistrés en dehors des rejouables, formant ainsi une « bibliothèque » partageable qui a la responsabilité, lorsqu'un représentant d'un événement est enrichi dans un rejouable, d'enrichir ce représentant dans chaque rejouable où il est présent. C'est à dire de rajouter ou mettre à jour la propriété de cet événement correspondant à cet enrichissement.

Cette modélisation est illustrée par la Figure 34. La trace devient un rejouable particulier qui enregistre un ensemble d'évènements observés, qu'il est possible d'enrichir par des relations (unaires pour les annotations, binaires pour les liens) qui leur rajoutent des propriétés supplémentaires. Les évènements dans les rejouables représentent des observés et par ce biais peuvent ajouter et récupérer des propriétés supplémentaires issues d'enrichissements.

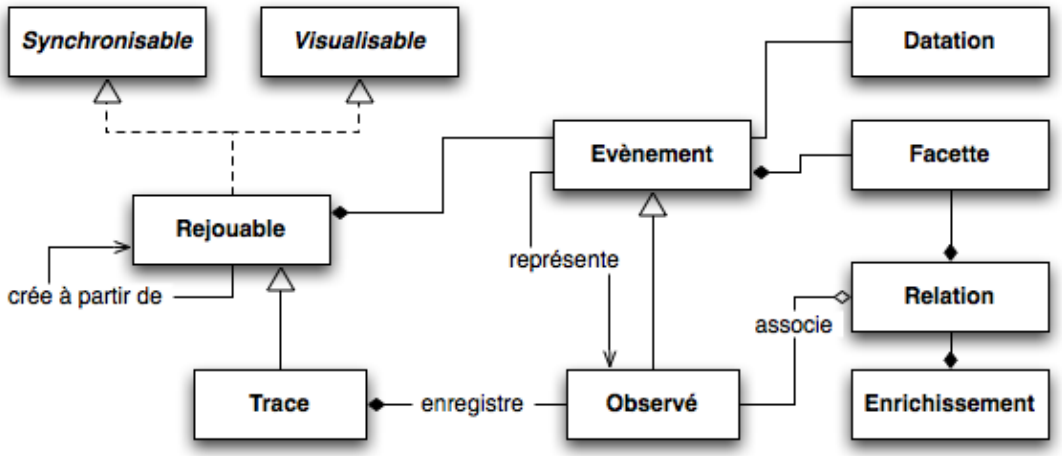


Figure 34 • Modélisation du rejouable et des entités liées dans le cadre de l'analyse.

Certains éléments d'analyse ne semblent pas être représentables par cette notion d'événement temporellement situé. Les objets comme les idées, les acteurs, les groupes peuvent cependant être représentés comme des évènements d'existence, qui commencent à la première apparition de l'objet et terminent lors de sa disparition. Tous les évènements en rapport avec cet objet peuvent alors voir « factoriser » la propriété « apprenant : Jean » en une relation entre ces évènements et un événement « Jean » qui durerait toute la séance. Les documents qui représentent un produit final ou une consigne et qui font partie du contexte de l'analyse sont plus problématiques. On peut en effet vouloir isoler des parties de document pour les mettre en lien avec des évènements de rejouable, mais on se retrouve alors avec une véritable collection d'objets (un document et toutes les sous-parties qu'on aurait identifiées) dont la situation temporelle n'est pas l'attribut contextuel principal. Cette problématique existe, mais l'examiner en plus de détail sort du cadre ce cet article.

Enfin, les données prises en « continu » ne sont pas de véritables évènements, puisqu'il n'y a pas de ponctualité ni d'événement à proprement parler, en dehors du choix de l'appareillage de collecte lui-même. Cela n'empêche que, hormis certains rares signaux (l'audio enregistré sur un vinyle, par exemple) ces signaux continus sont néanmoins enregistrés de manière numérique c'est à dire en effectuant des captures discrètes à intervalles réguliers. Pour autant que leur nature particulière est explicitée, on peut dès lors les traiter comme des évènements observés.

Propriétés des rejouables

Un *modèle* de rejouable va décrire, pour une instance particulière de rejouable, quels sont les types d'évènements qu'il peut contenir et, pour chaque type d'événement, quelles sont les

facettes qui le composent. Ce modèle pourrait être décrit à un niveau d'implémentation : types d'évènements, noms des facettes, types informatiques de ces facettes. Nous nous intéressons cependant au rôle de ce modèle pour l'analyse. En particulier, la transformation d'un rejouable en un autre va correspondre à la transformation d'un rejouable ayant un certain modèle en un rejouable ayant un autre modèle. Ce modèle va déterminer :

- les visualisations qui peuvent être faites ;
- l'utilité d'un rejouable pour un besoin analytique donné (par exemple, quel avantage à avoir une transcription plutôt qu'une vidéo ?);
- les transformations qui peuvent être faites ;
- le comportement de transformations génériques (par exemple une transformation générique de fusion de deux rejouables va créer un nouveau rejouable dont le modèle est différent – et calculable à partir des modèles précédents).

Nous ne nous attachons pas à décrire ce modèle de manière exhaustive ici, mais de dégager dans les grandes lignes les propriétés des rejouables qui sont pertinentes aux points évoqués ci-dessus. Nous adresserons dans une section subséquente le lien entre ces propriétés et la visualisation. Les propriétés des rejouables peuvent se situer à deux niveaux : celui des évènements qui le composent et celui des facettes qui décrivent ces évènements.

Au niveau des évènements, on peut distinguer les types d'échantillonnage : en continu (ou à intervalles réguliers non-liées à l'activité enregistrée) et de manière discrète (c'est à dire que les évènements crées sont liés à l'activité et provoqués par des caractéristiques de cette activité). L'échantillonnage discret peut être subdivisé en un échantillonnage dicté par le médium utilisé (action effectuée dans un environnement informatique, tour de parole) ou un échantillonnage choisi par l'analyste en fonction du cadre dans lequel l'analyse est effectuée (par exemple une activité, une étape, etc.). Ces changements de type d'échantillonnage correspondent à un besoin pour le chercheur d'établir un niveau de granularité adéquat pour son analyse.

Au niveau des facettes on peut distinguer différentes représentations : *numérique*, *textuelle*, *symbolique*, *identifiant* et *scalaire*. Toutes les facettes ont une représentation numérique qui nécessite la présence d'une machine pour la lire et la comprendre : on peut citer des exemples de vidéo (un rejouable dont les évènements sont des images), la trace de DREW qui, via son rejoueur, peut reproduire ce qui était affiché à l'écran ainsi qu'un enregistrement audio. La représentation textuelle est une représentation numérique qui utilise les conventions

informatiques (sa « machine ») qui permettent d'en extraire du texte. La représentation textuelle est directement lisible par un humain et peut être indexée afin d'y faire de la recherche d'information. La représentation symbolique est une représentation textuelle qui utilise un vocabulaire contrôlé auquel une sémantique supplémentaire et précise peut être attachée. Cette représentation ajoute la possibilité de dénombrer le nombre de fois que tel mot issu du vocabulaire a été utilisé. La représentation identifiante est une représentation symbolique dont chaque mot du vocabulaire correspond à une entité réelle (comme un apprenant, un artefact informatique, etc.). Enfin les représentations scalaires concernent les représentations numériques qui se traduisent en nombres qui peuvent aussi être immédiatement lues par le chercheur (pour autant que l'unité soit connue) ainsi qu'utilisée dans des requêtes utilisant des opérateurs d'égalité, mais aussi de comparaison.

Lors des transformations, notamment celles qui subsument plusieurs évènements en un seul, la représentation des différentes facettes peut nous dire comment ces facettes devrait se comporter lors de leur fusion avec un autre. Les représentations textuelles peuvent se concaténer, les représentations symboliques peuvent se regrouper dans un ensemble et les représentations scalaires peuvent se combiner selon différents opérateurs (somme, maximum, etc.).

Enfin, toujours au niveau des facettes, on peut distinguer celles qui sont issues de la trace, celles qui sont calculées à partir de la trace ainsi que celles qui sont rajoutées par le chercheur. La provenance d'une facette va informer son niveau de « subjectivité » et permettre au chercheur de mieux l'évaluer lors de son exploitation. Il va éventuellement vouloir la valider (par exemple en faisant intervenir une deuxième personne ou en examinant les données source de plus près).

Visualisation de rejouables

Les visualisations de rejouable se différencient par leur représentation du temps ainsi que par leur capacité de traitement des différents types de rejouables qu'ils représentent. Les propriétés d'une visualisation vont contribuer à sa pertinence lors de l'analyse.

Lorsque le temps n'est pas représenté dans la visualisation, il faut user du rejouage pour rétablir cet aspect temporel. C'est le cas pour l'audio, la vidéo, ainsi que pour les visualisations ou la proximité spatiale n'est pas indicatrice de proximité temporelle (par exemple un graphe conceptuel pourra être parcouru dans l'ordre d'apparition des éléments).

Les rejouables composés d'évènements ayant des facettes à représentation numérique posent un problème à ce niveau : leur visualisation se fait le plus souvent par des images en deux dimensions, rendant impossible d'y représenter le temps. Le temps d'analyse est alors augmenté par la nécessité de rejouer pour resituer le contexte temporel. Il devient par ailleurs difficile de représenter d'autres facettes dans la même visualisation.

Les tableaux, lignes et courbes temporelles ainsi que les textes représentent le temps selon un de leurs axes. Ils peuvent aussi simuler le rejouage et assister à la synchronisation en mettant en évidence le passage du temps sur cet axe (sélection d'une ligne dans un tableau ou d'un objet graphique dans une ligne de temps). Le tableau présente certaines restrictions par rapport à la présentation d'évènements qui se chevauchent, qui peuvent être représentés par des lignes différentes dans une ligne de temps, un graphe ou une transcription

Lors de l'absence de chevauchements entre évènements, les tableaux ont l'avantage de pouvoir représenter un grand nombre de facettes (une facette par colonne), ainsi que de présenter facilement les facettes à représentation scalaire et textuelle. Les courbes permettent de prendre compte facilement de données scalaires. Les lignes de temps sont bien adaptés à la présentation succincte de facettes symboliques et identifiantes, chaque mot du vocabulaire pouvant être associé à une propriété graphique comme la forme, la couleur ou la position verticale (si la position horizontale est déterminée par le temps). Elles sont aussi bien adaptées à la visualisation de relations entre événements, contrairement aux tableaux qui, même dans le cas de tableaux arborescents ne peuvent que présenter des graphes acycliques.

Transformations de rejouable

Afin d'illustrer ces propriétés de rejouables ainsi que de leurs visualisations, nous pouvons examiner quelques transformations et l'effet qu'elles ont. Cet effet peut éclairer l'intention du chercheur ayant effectué cette opération.

La *transcription* est la transformation d'un rejouable à échantillonnage continu et n'ayant que des facettes à représentation numérique nécessitant un lecteur audio (ou audio-video) pour les comprendre en un rejouable à échantillonnage discret, ayant une facette identifiante (locuteur) ainsi qu'une facette textuelle (énoncé) qui peut être visualisé sans rejouage. Le coût très grand de la transcription peut alors être évalué en fonction des nombreux gains en temps qui seront possibles avec son rejouable résultant (visualisation, recherche, calculs). Les étapes de

segmentation, de transcription textuelle et d'*identification* du locuteur pourraient être séparées ou seulement effectuées de manière partielle, en fonction des besoins subséquents.

Le *filtrage* est la transformation d'un rejouable dont les évènements ont au moins une facette symbolique en un rejouable n'ayant que les évènements dont cette facette prend une valeur donnée. Il peut être nécessaire de procéder au préalable à une transformation d'*identification* afin d'identifier les objets, acteurs ou idées sur lesquelles on pourrait vouloir focaliser, ou a une *catégorisation*, qui rajoute une facette symbolique à chaque élément. La *recherche* est similaire au filtrage, mais s'applique à une facette textuelle et peut faire intervenir la notion de similarité entre le contenu de cette facette et la requête. La création d'une collection est une forme de filtrage ou le rejouable résultant ne contient que des évènements ayant une caractéristique commune. Si cette opération est effectuée à la main, la présence de cette facette symbolique peut être implicite. Il peut alors être utile de l'expliciter afin de l'exploiter dans des filtrages complexes.

La *fusion* combine les évènements issus de différents rejouables sans les transformer. Elle permet d'avoir ces évènements dans une seule et même visualisation, sans avoir à recourir à la synchronisation. Lors d'une fusion, il peut être utile de s'assurer de la cohérence entre les facettes des différents types d'évènements. Les facettes à représentation scalaire doivent être de même unités lorsque cela est possible. Les facettes à représentation symbolique doivent assurer que le même vocabulaire contrôlé est utilisé pour représenter les mêmes choses, particulièrement lorsque cette représentation symbolique est identifiante : par exemple si une transcription utilise les noms *A* et *B* pour désigner de manière anonyme les participants et que la trace numérique utilise leurs pseudos non-anonymisés, il convient d'unifier ces représentations et utiliser *A* et *B* partout.

Enfin, le *regroupement* rassemble plusieurs évènements d'un rejouable en un seul événement. Les valeurs des facettes résultant d'un regroupement peuvent parfois être automatiquement déduites. De même, ce nouvel événement peut être considéré comme le représentant de l'ensemble des observés représentés par les évènements qu'il regroupe. L'exploitation des enrichissements peut alors persister en travers un regroupement.

Application

Cette modélisation du rejouable est le fruit de notre réflexion après une première implémentation d'un environnement de manipulation de rejouables, nommé Tatiana (*Trace*

Analysis Tool for Interaction ANAlysts). Nous sommes donc en mesure de présenter un exemple d'analyse qui exploite pleinement les possibilités d'un tel environnement ainsi que de dégager quelques problématiques d'implémentation.

Exemple sur une analyse de reformulation

Nous avons observé neuf dyades sur une série de trois à cinq rendez-vous avec leur encadrant, dans le contexte du projet de programmation d'un cours d'introduction à l'informatique. Ces réunions se déroulaient en face à face, avec utilisation d'un « chat » et d'un éditeur de texte partagé, auxquels les participants avaient accès sur des ordinateurs portables. Les étudiants étaient incités à prendre leurs notes dans l'éditeur de texte partagé. Les traces collectées lors de ces séances sont une vidéo pour enregistrer les gestes et le dialogue oral ainsi que les traces numériques du chat et de l'éditeur de texte partagé. La vidéo, comme nous l'avons vu, est un rejouable présentant des évènements en échantillonnage continu et dont la seule facette (l'image capturée à chaque instant) nécessite une machine pour traduire sa représentation. La trace du *chat* est un rejouable dont les évènements sont à échantillonnage discret, produits chaque fois qu'un utilisateur envoie un message. Les facettes d'un événement de chat sont un nom d'utilisateur ainsi (représentation identifiante) ainsi que le texte rédigé (représentation textuelle). La trace de l'éditeur de texte partagé est un rejouable dont les évènements sont à échantillonnage continu (toutes les secondes, pourvu qu'un changement se soit produit) dont les deux facettes sont un nom d'utilisateur (représentation identifiante) et le contenu – le texte complet courant – de l'éditeur de texte partagé (représentation textuelle). Cette facette nous informe sur l'état de l'éditeur de texte, mais ne nous dit rien sur la différence entre cet état et l'état précédent. Cette trace est donc le plus aisément compréhensible grâce à une machine : le rejoueur (de DREW).

Lors de l'une des analyses, nous nous sommes intéressés aux reformulations d'un médium à un autre, en particulier du discours oral aux notes dans l'éditeur de texte partagé (Dyke *et al*., 2007). Nous voulions aboutir à une meilleure compréhension de ces reformulations d'un point de vue linguistique, et en explorer les conséquences pédagogiques éventuelles. Pour parvenir à cette fin il nous fallait un rejouable (ou un ensemble de rejouables) qui représente des unités d'expression médiatique à une granularité telle qu'un lien de reformulation fasse sens, permettant alors d'expliciter ces liens et de les examiner en plus de détail. Pour le chat (peu utilisé), nous avons laissé cette granularité au niveau du message. Pour le dialogue, nous l'avons mis au niveau du tour de parole (avec des coupures lors de pauses pendant les longs

tours de parole) en effectuant une transcription manuelle. Pour l'éditeur de texte partagé, en revanche, ils nous a fallu établir la notion *d'unité de rédaction* qui, informellement, consiste en une clause ou phrase dans les notes rédigées qui forme une unité sémantique. Afin de transformer la trace de l'éditeur de texte partagé en rejouable contenant des unités de rédaction, nous sommes passés par un rejouable contenant les mêmes évènements mais considérant ces évènements comme étant des actions et en calculant une facette pour chaque événement qui décrive le changement par rapport à l'état précédent, puis par un rejouable crée au travers d'une segmentation automatique basée sur un temps d'inactivité donné. Ce rejouable à assisté une segmentation manuelle en unités de rédaction ainsi que l'annotation de cette segmentation avec des facettes pour décrire le texte rédigé ainsi que le rédacteur pendant cette unité de rédaction. Afin d'avoir un rejouable sur lequel visualiser les liens de rédaction, nous avons fusionnée la transcription et les unités de rédaction. Des visualisations de ces différents rejouables sont illustrées dans la Figure 35. On y retrouve les deux traces visualisées dans un lecteur vidéo et dans le rejoueur de DREW. Le premier résultat de transformation de la trace de l'éditeur de texte partagé (montrant la différence d'un état à l'autre), le rejouable représentant les reformulations ainsi que la transcription ont un contenu principalement textuel et sont visualisés sous forme tabulaire. Le rejouable issu de la fusion entre les unités de rédaction et la transcription est visualisé sous forme de ligne du temps, d'une part pour mieux comprendre les chevauchements et d'autre part afin de visualiser les liens de reformulation (représentés à la fois par des couleurs identiques et des flèches).

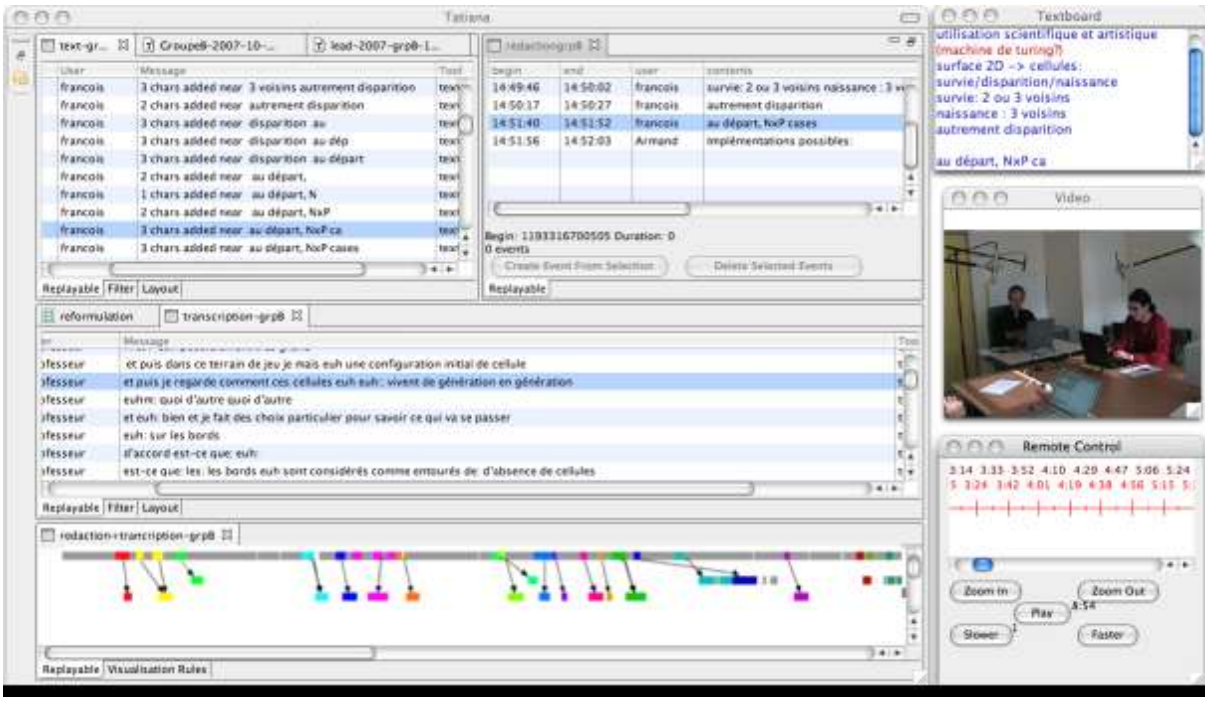


Figure 35 • Visualisation synchronisée de différents rejouables dans Tatiana. En haut à gauche : Trace de l'éditeur de texte partagé et vidéo. Au milieu : rejouables issus de la trace de l'éditeur de texte partagé ; traduction directe de la trace primaire et unités de rédaction. Au milieu : transcription de la vidéo. En bas : Fusion de la transcription (ligne du haut) et des unités de rédaction (une ligne par apprenant) enrichis des liens de reformulation.

Difficultés d'implémentation et limites du rejouable

Les majeures tensions d'implémentation se font sentir lors de la gestion de la dualité entre l'automatique et le manuel. Nous avons souhaité qu'un maximum de choses soient automatisées : la synchronisation est automatique, tout comme les visualisations (bien que ces dernières soient paramétrées par des feuilles de style définies par l'utilisateur). Certaines transformations (par exemple fusion, calcul sur la trace, recherche) sont automatiques, d'autres (segmentation, regroupement et re-ordonnancement d'évènements) sont manuelles. Cela nous a conduit à différencier les rejouables issus de ces deux types de transformation. En effet, si une source de transformation automatique change (par exemple parce qu'elle est construite à la main), le résultat est automatiquement mis à jour. En revanche, les transformations manuelles ne sont pas (pour l'instant) automatiquement re-appliquées lorsque le rejouable source change. Dans une problématique similaire, les regroupements manuels n'appliquent pas automatiquement l'union des facettes des évènements composants mais proposent plutôt des facettes (représentés par des champs vides dans un tableau) que l'utilisateur doit remplir à la main. Enfin, les enrichissements ne persistent pas sur les transformations de regroupement.

Ces différents problèmes viennent entre autre du fait que nos rejouables n'ont pas de modèle explicite permettant de déterminer que faire d'une facette lors du regroupement. Cela nous a permis de revisiter notre modélisation du rejouable pour définir les différentes représentations de facette. Notre choix de ne pas associer nos traces à un modèle explicite est pourtant le résultat d'un choix motivé : la définition explicite de modèles est un coût supplémentaire lors de l'analyse. Cette analyse étant itérative, elle emprunte de nombreux chemins qui n'ont pas de débouché final avant d'aboutir à un ensemble de rejouables suffisants à l'interprétation. Si le chercheur devait, à chaque étape, définir le modèle du rejouable, puis l'ajuster à chaque itération, le coût engendré pourrait pousser le chercheur à limiter son exploration libre des données et par conséquent restreindre la flexibilité de l'outil. La prochaine étape nous paraît

donc être, au prix d'une modélisation minimale des traces d'origine, d'introduire une méthode de calcul explicitant les modèles des rejouables subséquents, basée sur les types de représentation des facettes.

Une autre problématique vient de l'abstraction de notre modélisation du rejouable. Il devient possible d'instancier le même besoin analytique de différentes façons qui sont presque – mais pas entièrement – équivalentes. Par exemple, supposons que nous avons une transcription manuelle dans Tatiana. Cette transcription est un rejouable dont la segmentation des évènements et le remplissage des facettes sont faits à la main. Le chercheur est libre de définir les facettes qui seront contenues dans ce rejouable. Supposons maintenant que le chercheur veuille isoler certains de ces évènements. Il existe trois façons de définir cette isolation : par une transformation manuelle sélectionnant les éléments voulus ; par une transformation rajoutant une facette au rejouable qui indique une propriété supplémentaire puis en appliquant un filtre sur cette propriété ; en rajoutant une facette au rejouable via un enrichissement puis un filtre sur cet enrichissement. De manière plus générale, on peut se poser les deux questions suivantes : dois-je rajouter une propriété aux évènements d'un rejouable par la création d'un nouveau rejouable dont les évènements ont cette propriété supplémentaire ou avec un enrichissement ? dois-je regrouper des éléments dans un nouveau rejouable simplement en les sélectionnant à la main ou rajouter un enrichissement et appliquer un filtre ? La réponse à ces questions est peut-être simplement d'ordre ergonomique : si on veut profiter de l'intégration automatique d'une propriété sur l'arbre des rejouables, il faut faire un enrichissement. Si cela n'est pas nécessaire ou désirable, on peut faire autrement.

L'intégration des enrichissements d'évènements sur l'arbre des rejouables dépend de la possibilité d'identifier les observables que représentent ces évènements. Vu que la co-occurrence de deux observables différents sur le même intervalle de temps n'est pas à exclure, nous ne pouvons limiter l'identification à un datation. Pour les observables présents dans les traces à échantillonnage discret, cette identification est aisée. En revanche, pour les observables présents dans les traces à échantillonnage continu, il peut être nécessaire de repréciser la datation d'un événement (un geste, par exemple), de distinguer deux évènements différents sur le même intervalle ou encore de traiter comme identiques deux évènements dont les extrémités d'intervalle sont légèrement différentes. Pour l'instant, nous nous contentons d'identifier de tels évènements par un nom de trace et un intervalle de temps.

Le dernier problème que nous avons relevé concerne les limites de la synchronisation temporelle. En effet et comme nous venons de le soulever, co-occurrence n'implique pas

identité. Par exemple, à la Figure 35, la sélection d'une unité de rédaction va mettre en évidence le tour de parole co-temporel dans la transcription. Or, cette unité de rédaction est souvent la reformulation d'un tout autre tour de parole et la synchronisation peut prêter à confusion.

Conclusion

Dans cet article, nous avons présenté les besoins d'analyse dans le cadre du CSCL et des domaines proches. Nous avons montré la nécessité de la définition d'un objet d'analyse qui puisse satisfaire ces besoins et servir de base pour l'implémentation d'outils d'analyse ainsi que de dialogue entre chercheurs issus de différentes épistémologies. Nous avons proposé la notion de *rejouable*, objet abstrait qui peut être synchronisé, visualisé, transformé et enrichi. Nous avons ensuite examiné les propriétés de l'objet proposé vis-à-vis des opérations dont nous l'avons doté. La modélisation de rejouable que nous avons proposée est le retour après une première implémentation dont nous avons isolé les difficultés.

Cette modélisation devra être mise à l'épreuve dans le cadre d'analyses à grande échelle (de nombreuses analyses sont actuellement en cours avec le logiciel Tatiana). Nous avons mis en évidence le besoin d'associer à chaque rejouable un modèle qui permette de mieux définir les visualisations et transformations à partir de ce rejouable. Nos travaux subséquents porteront sur le calcul implicite de ce modèle afin de ne pas imposer au chercheur une définition de modèles fastidieuse.

Notre modélisation du rejouable répond aux problématiques de morceaux, codes, commentaires, connexions, conversions et contraintes définies par Sanderson et Fischer. Les calculs créent des objets qui ne sont plus des rejouables mais qui sont sans doute dignes d'étude. Enfin, les comparaisons peuvent en partie être faites grâce à la réification des rejouables et des enrichissements en objets informatiques partageables et visualisables en parallèle.

Cela nous amène à la problématique plus large du passage à l'échelle des analyses en CSCL qui devra passer par un partage de corpus, un dialogue entre différentes épistémologies ainsi qu'une utilisation d'outils d'analyse qui permettent la confrontation et la complémentarité de méthodologies d'analyse multiples. Nous espérons que la notion de rejouable permettra d'effectuer des avancées sur chacun de ces points.

1. Introduction

L'étude sociocognitive des interactions humaines médiatisées par ordinateur peut s'effectuer au travers de l'enregistrement de ces activités, en particulier si elles ne sont pas limitées aux seules traces produites par les outils, mais incluent également des enregistrements audio et vidéo de ces activités (Avouris *et al.*, 2007). Cox (2007) encourage les chercheurs à utiliser l'ordinateur et les diverses techniques qu'il propose (visualisation, data mining, etc.) pour effectuer leurs analyses des interactions dans ce qu'il appelle « données du processus » (« *process data* »). Cependant, les corpus d'interaction, en particulier lorsque ces interactions médiatisées se déroulent en face à face, sont difficiles à gérer d'un point de vue technique, et compliquées à appréhender du fait de la multiplicité et de la diversité des sources de données. Il ne suffit pas en effet d'analyser des flots de données séparés ; ces flots de données doivent être combinés pour permettre une compréhension globale des interactions qui ont eu lieu (Goodman *et al.*, 2006). Qui plus est, ces analyses doivent souvent être effectuées en équipe, que ce soit pour valider la méthode d'analyse au travers de la méthode des juges (De Wever *et al.*, 2006), pour répartir la charge de travail (Goodman et al., 2006), ou pour combiner les perspicacités de plusieurs chercheurs de domaines différents (Prudhomme *et al.*, 2007).

Les difficultés décrites ci-dessus impliquent la nécessité d'outils qui puissent non seulement gérer cette variété et quantité de données, mais aussi permettre la visualisation et l'analyse dans un cadre de travail commun qui puisse être partagé avec d'autres chercheurs (cf. Reffay *et al.*, 2008 pour le travail de structuration de corpus d'apprentissage dans un but d'échange et de réutilisation ainsi que Suthers *et al.*, 2008 pour le besoin d'avoir des outils pour gérer le partage d'analyses). Les outils existants (par exemple, Greenhalgh *et al.*, 2007 et Fiotakis *et al.*, 2007) montrent les possibilités de l'analyse assistée par ordinateur mais sont spécialisés à certains types de données ou à certains types d'analyse et nécessitent des développements supplémentaires pour fonctionner de manière interopérable (Kahrimanis *et al.*, 2006). Nous proposons la création d'un outil d'aide à l'analyse qui soit générique, extensible et qui puisse être utilisé en combinaison avec d'autres outils.

Dans cet article, nous donnons un aperçu de quelques-unes des opérations effectuées par les chercheurs qui analysent les traces d'interactivités entre individus, en particulier lorsque ces activités sont médiatisées par ordinateur (nous nous concentrerons sur notre domaine

d'application, le CSCL[30], mais ces pratiques s'étendent à d'autres domaines des STIC, notamment le CSCW[31]). Nous présentons ensuite un modèle simple de description de ces activités, et montrons comment le logiciel Tatiana (*Trace Analysis Tool for Interaction ANAlysis*) a été conçu pour soutenir ce modèle, que nous considérons comme généralisable à d'autres types d'analyses. Plus spécifiquement, nous montrons comment Tatiana peut aider les chercheurs à réaliser les opérations que nous avons présentées. Nous terminons enfin en montrant comment Tatiana est conçu pour s'adapter à de nouvelles formes d'analyses, et encourage le partage et la collaboration dans les recherches sur l'interaction humaine.

2. Comment se déroule l'analyse ?

Un corpus typique d'interactions médiatisées par ordinateur peut contenir des enregistrements audio et vidéo, de fichiers de traces d'interactions médiatisées par ordinateur, des interviews, des notes de terrain, des tests précédant et suivant l'expérimentation, et la description de celle-ci (finalité, contexte, scénario, etc.). Lors de la conception de notre outil destiné à permettre l'analyse d'un tel corpus, nous avons tenté de répondre à la question suivante : « que font les chercheurs de ces données ? »

Bien que cette question puisse être abordée sur le plan méthodologique, parvenir à un recensement raisonnable des pratiques existantes dans le domaine du CSCL (et de celles de communautés proches) aurait conduit à mener une méta-analyse d'un très vaste ensemble de recherches ; le faire d'une manière satisfaisante déborderait largement du cadre de cet article. De plus, nous pourrions nous livrer à une telle analyse, mais elle montrerait ce qu'*obtiennent* les chercheurs, et pas nécessairement de *quelle manière* ils l'obtiennent, dans la mesure où cette démarche est très rarement décrite dans la littérature. En tant que concepteurs d'un outil destiné à aider les analyses, il nous a semblé difficile de recueillir cette information sur une large échelle. Dans cette section, nous présentons notre compréhension de l'activité d'analyse en explorant un sous-ensemble des thèmes d'analyse et certains des artéfacts que créent les chercheurs en liaison avec ces thèmes. Nous nous appuierons, pour étayer nos thèses, sur des cas d'études, sur des articles décrivant des questions ou des expérimentations méthodologiques, ainsi que sur les outils existants, destinés à aider l'analyse. Enfin, nous présentons le modèle de l'analyse sur lequel nous avons basé la conception de Tatiana.

[30]. *Computer-Supported Collaborative Learning* ou Apprentissage collaboratif médiatisé par ordinateur.

[31]. *Computer-Supported Collaborative Work* ou Travail collaboratif médiatisé par ordinateur.

2.1. Etudes de cas

Dans le contexte du projet Lead[32] (Lead, 2006), nous avons eu l'opportunité d'observer les activités de recherche de quatre équipes lors de la conception et de l'analyse d'expérimentations liées à ce projet. Toutes ces expérimentations ont pour objet l'utilisation en présentiel de communications médiatisées par ordinateur (les communications sont verbales ou médiatisées par ordinateur). L'analyse, dans ces conditions, présente des problèmes particuliers (Dyke *et al.*, 2007), l'un d'eux étant de parvenir à une synchronisation du rejouage des données disponibles dans les fichiers de traces du logiciel utilisé et des données audio et vidéos enregistrées, afin de bien comprendre la situation. Nous avons pu travailler en étroite collaboration avec l'une des équipes, et nous avons pu nous rendre in situ auprès de deux autres équipes, où nous avons enregistré et discuté leurs activités d'analyse. En présentant ces études de cas, nous ne pouvons entrer en détail sur les hypothèses et les suppositions théoriques, qui sont liées à des études non encore publiées. Nous donnerons un aperçu de la procédure suivie par chaque équipe pour une partie de leurs travaux, en montrant que leur approche valide notre vision sur la manière dont l'analyse est effectuée par les chercheurs. Le développement de Tatiana, effectué en parallèle, a permis d'effectuer certaines analyses grâce à cet outil, mais, en particulier lors des visites auprès de ces équipes, nous avons découvert des besoins que nous ne pouvions alors satisfaire. Dans cette section, nous nous intéressons aux étapes effectives des analyses que les équipes désiraient accomplir, indépendamment de l'usage éventuel de Tatiana.

2.1.1. Etude de cas, Lyon/Saint-Étienne

Nous avons été les premiers utilisateurs de Tatiana durant les phases initiales du développement (avec, à certains moments, la casquette du développeur, à d'autres celle du chercheur en CSCL ou du testeur de l'usabilité de l'outil). Nous avons observé neuf dyades sur une série de trois à cinq rendez-vous avec leur encadrant, dans le contexte du projet de programmation d'un cours d'introduction à l'informatique. Ces réunions se déroulaient en face à face, avec utilisation d'un « chat » et d'un éditeur de texte partagé, auxquels les intervenants avaient accès sur des ordinateurs portables. Les étudiants étaient incités à prendre leurs notes dans l'éditeur de texte partagé.

[32]. Le projet Européen LEAD (*Technology-enhanced learning and problem-solving discussions: Networked learning environments in the classroom,* http://www.lead2learning.org/) est financé par le 6ème framework *Information Society Technology* LEAD IST-028027.

Lors de l'une des analyses, nous nous sommes intéressés à la provenance de ces notes, qui étaient très souvent des reformulations d'interventions orales. Nous voulions aboutir à une meilleure compréhension de ces reformulations d'un point de vue linguistique, et en explorer les conséquences pédagogiques éventuelles. Pour parvenir à ces fins, nous avons d'abord effectué une transcription des dialogues. Nous avons également transformé les traces de l'éditeur de texte partagé pour que ces traces fassent sens pour le chercheur. L'implémentation de cet éditeur de texte partagé est telle que des « évènements » sont enregistrés chaque seconde, contenant le texte complet, le nom de l'intervenant et la position du curseur. Nous voulions procéder à des analyses de plus haut niveau, et donc disposer d'informations contenant plus de sémantique. Ces évènements ont donc été regroupés en unités de rédaction, nous permettant alors d'établir les liens de reformulation entre les transcriptions et ces unités de rédaction. Enfin, les évènements ont été visualisés (cf. Figure 1), ce qui a permis de mettre en évidence certains phénomènes intéressants, et d'acquérir une compréhension intuitive du processus de reformulation.

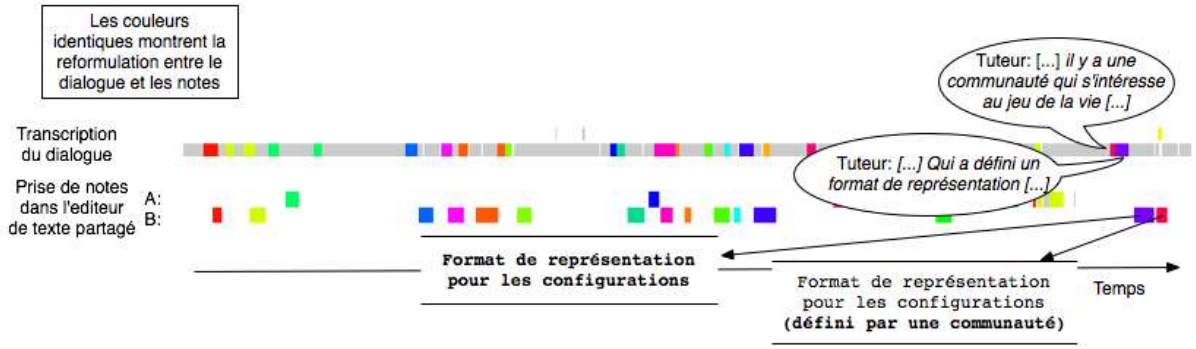


Figure 36. Un exemple de visualisation de reformulation

2.1.2. Etude de cas, Paris

Dans ce second cas, nous nous sommes rendus sur place pendant deux jours, pour comprendre les pratiques de l'équipe et examiner comment Tatiana pouvait s'avérer utile dans le cadre de ces pratiques. Le résultat fut matérialisé par l'enregistrement audio et vidéo des discussions et d'une session de test de Tatiana, que nous ne détaillerons pas ici. L'une de leurs analyses impliquait l'étude, sur le long terme, de l'utilisation du logiciel CoFFEE (De Chiara *et al.*, 2007), développé durant le projet LEAD. Il s'agissait d'un suivi de paires de dyades travaillant en commun dans un espace partagé. Chaque dyade disposait d'une machine, et chaque paire de dyade travaillait dans l'espace commun, les deux machines des deux dyades étant placées côte à côte. L'équipe s'intéressait, dans cette analyse, aux types d'interactions

(Bernard, 2008) se produisant au sein d'une dyade (focus sur l'outil ou sur le sujet du travail, communication orale ou médiatisée, etc.), en souhaitant découvrir le lien entre ces interactions et la collaboration entre les dyades.

Pour procéder à cette étude, ils désiraient jouer la vidéo en synchronisation avec les traces médiatisées, en opérant un marquage des blocs tandis que les différents types d'interaction se produisaient au sein de chaque dyade. Ces interactions seraient alors catégorisées selon un schéma de codification pré-établi, décrivant les différents types d'interaction, et permettant la comparaison en parallèle des interactions de chaque paire de dyade avec les interactions médiatisées.

2.1.3. Etude de cas, Utrecht

Nous avons également passé deux jours in situ pour ce troisième cas, analysant les pratiques du groupe pour déterminer si Tatiana était adapté dans ce cas. Ce séjour se traduisit également par l'enregistrement vidéo et audio des discussions. Leur travail s'intéresse principalement à la production de diagrammes argumentatifs (c.f. Overdijk *et al.*, 2008). Une de leurs méthodes d'analyse consiste à conserver les seuls évènements consacrés à la création de boîtes, puis à réordonner ces évènements pour faire apparaître la structure du diagramme (à la manière des fils de discussion d'un forum) plutôt que leur enchaînement chronologique. A partir de cette production, ils peuvent appliquer sur les données les méthodes d'analyse de discours ou de contenu.

Un autre processus d'analyse utilisé est la création de visualisations qui montrent comment l'activité des étudiants se déploie dans les différents fils de discussion au cours du temps. Ces vues combinées (chronologiques, par fil de discussion, par étudiant) offrent une bonne visibilité et permettent une meilleure compréhension de la situation (Lund et al., 2008).

2.1.4. Etude de cas, Nottingham

Dans ce dernier cas, nous n'avons pas eu l'opportunité d'effectuer de visite sur le site, mais, comme avec les autres partenaires, nous avons pu avoir de nombreuses discussions au cours des réunions du projet LEAD ou par email. Une de leurs pratiques consiste à utiliser des méthodes statistiques (telles que celles qui sont disponibles dans SPSS ou Excel) au cours de leurs analyses. Dans une de leurs études (faisant suite à Gelmini Hornsby *et al.*, 2008), ils étaient intéressés à comprendre comment différentes circonstances influaient sur les changements d'opinions exprimées dans le module de vote de CoFFEE. Le scénario

comportait un premier vote, une discussion, puis un second vote. Avant la fermeture de chaque scrutin, les étudiants pouvaient changer d'opinion autant de fois qu'ils le désiraient, ce qui était enregistré dans les traces du logiciel. Les chercheurs devaient chercher manuellement le dernier changement d'opinion pour chaque étudiant et chaque vote dans les traces, afin de disposer des données nécessaires à leur étude.

Ils ont aussi exprimé un intérêt particulier pour le calcul automatique d'indicateurs statistiques assez généraux en relation avec les discussions, tels que le nombre de mots par tour de parole, par tour et par étudiant, le nombre d'interventions par étudiant et par sujet, etc. La raison d'être de ces demandes était que tout phénomène inaccoutumé (par exemple, une participation inhabituellement forte ou faible) pouvait être un point de départ pour une analyse plus poussée. Ils ont également confirmé l'importance, pour tout nouvel outil, de pouvoir s'intégrer à leurs pratiques habituelles (utilisation de Excel, SPSS, mais aussi d'autres outils de transcription ou d'analyse).

2.2. Quelques thèmes d'analyse

Comme nous l'avons déjà mentionné, notre but n'est pas de couvrir la totalité (ni même la majorité) des méthodes d'analyse, de leurs présupposés théoriques et des complexités de leurs conditions d'application. Au contraire, nous avons étudié ces approches de l'analyse pour mettre en évidence, sur le plan pragmatique, les types d'opérations que les chercheurs sont amenés à effectuer.

2.2.1. Codage et Comptage

Plusieurs types d'analyses utilisent le codage et le comptage, le plus souvent pour des analyses de contenu (Strijbos *et al.*, 2006). Cette pratique est tellement commune qu'il existe une importante littérature autour des techniques variées d'intelligence artificielle destinées à aider à automatiser cette tâche de comptage (Rosé *et al.*, 2008; Erkens *et al.*, 2008). Dans l'application de ce type de méthodes, une question importante est celle de l'unité d'analyse (De Wever *et al.*, 2006), à savoir la granularité pertinente : est-ce le tour de dialogue, une proposition au sein d'un tour de dialogue, une succession de tours de dialogue ? Une autre question est celle du codage à appliquer, qui peut être tiré de la littérature (e.g. Baker *et al.*, 2007) ou être spécifiquement créé pour une analyse particulière. Une fois ce schéma appliqué

se pose la question de sa validation (De Wever *et al.*, 2006). Enfin, il convient d'effectuer les calculs statistiques désirés.

Quel que soit le choix de l'unité d'analyse, le corpus doit être découpé en de telles unités. Selon la nature du médium, différentes segmentations sont possibles. La transcription des dialogues et des gestes humains à partir d'un support audio ou vidéo s'effectue typiquement avec un outil tel que Elan (http://www.lat-mpi.eu/tools/elan/). Les traces des outils informatiques sont transformées en séquences d'actions, qui peuvent alors être regroupées, re-segmentées ou filtrées (dans le cas d'étude de Nottingham on pourrait sélectionner que le dernier vote de chaque élève). Dans tous les cas, le résultat se présente sous formes d'évènements (ou d'actions), dotés de propriétés telle que la date, l'outil, l'utilisateur, le contenu échangé, etc. Plus concrètement, ces données peuvent se représenter dans un outil tel qu'Excel™, avec une ligne pour chaque évènement et une colonne pour chaque propriété. Parmi les logiciels de support à l'analyse, Elan, Videograph® (http://www.ipn.uni-kiel.de/aktuell/videograph/enhtmStart.htm) et DRS (Digital Replay System, Greenhalgh *et al.*, 2007) permettent directement l'annotation et le codage des vidéos d'une manière convenable pour le cas d'étude parisien.

Les schémas de codage et d'évaluation sont de natures très diverses. Ils sont souvent le résultat d'un processus itératif impliquant un codage partiel du corpus, puis une redéfinition du schéma de codage, voire un travail collaboratif de codage (De Vries *et al.*, 2002). Il est rare qu'un schéma de codage préexistant soit appliqué sans aucune modification. Cet aspect est important pour la conception du logiciel, puisqu'il implique de devoir prendre en compte les codages selon des catégorisations évolutives.

Le codage peut dès lors être considéré comme l'ajout, par la personne effectuant l'analyse, d'une ou de plusieurs propriétés à chaque évènement. La nature subjective de cette activité, ainsi que la nécessité d'effectuer ultérieurement des analyses statistiques sur les données codées impliquent une validation de ce codage. Il est fréquent de voir appliquer la méthode des juges (De Wever *et al.*, 2006), qui, indépendamment de la métrique utilisée (le kappa de Cohen, l'alpha de Krippendorf, etc.), implique de comparer côte à côte au moins deux codages, qu'ils soient complets ou partiels.

Les données sont alors regroupées en fonction des codes attribués et de certaines autres caractéristiques (comme l'utilisateur, le groupe, l'outil, etc.). Par l'observation des distributions obtenues, et par l'application d'outils statistiques, certaines hypothèses peuvent être confirmées ou infirmées.

2.2.2. *Signets, collections et annotations*

D'autres formes d'analyse, notamment les études de cas (ex. Rummel *et al.*, 2008), l'analyse conversationnelle (Sacks *et al.*, 1974) et les analyses basées sur la théorie des activités (ex. Avouris *et al.*, 2007), consistent à décrire le corpus par l'ajout d'annotations, la création de collections ou de groupes d'évènements. Ces opérations peuvent s'effectuer manuellement (dans un fichier texte, ou avec un tableur), ou encore au moyen d'outils dédiés, comme ActivityLens (Fiotakis *et al.*, 2007), Videograph ou Transana (http://www.transana.org/).

2.2.3 Synchronisation temporelle

La nécessité de synchroniser les différentes sources de données ainsi que les artéfacts construits par le chercheur est évidente : les évènements observés — au moment de l'expérimentation — sont situés temporellement, et doivent être replacés dans leur contexte afin de pouvoir être compris. De plus, les différentes vues des mêmes évènements (p.ex. vidéo et transcription) se complémentent bien lorsqu'elles sont synchronisées. En fait, c'est également une nécessité scientifique que de disposer de cette synchronisation (ou au minimum de pouvoir se reporter aux données primaires) : lorsque l'on cherche à établir une hypothèse, la question qui se pose systématiquement est : « dispose-t-on dans les données primaires d'éléments soutenant cette hypothèse ? ». Les chercheurs ont besoin d'artéfacts qui présentent le corpus sous une forme mieux adaptée à la compréhension, l'affichage ou l'analyse (comparons par exemple des données audio ou vidéo et leur transcription). Cependant, quand une hypothèse est avancée à partir d'informations découvertes dans de telles données secondaires, il est nécessaire de vérifier que l'on retrouve effectivement dans les données originelles la confirmation de cette hypothèse. La synchronisation offre un moyen de revenir aisément aux données primaires chaque fois qu'une telle confirmation est nécessaire.

La synchronisation des différents média sources est prise en compte, à des degrés divers, par nombre d'outils : Elan, ActivityLens, DRS, Replayer (Morrison *et al.*, 2006) et ABSTRACT (Analysis of Behaviour and Situation for menTal Representation Assessment and Cognitive acTivity modeling; Georgeon *et al.*, 2007; Georgeon *et al.*, 2006) permettent tous la synchronisation des artéfacts qu'ils construisent (c.f. Figure 37).

Figure 37. Dans cet exemple du logiciel Replayer, les évènements sélectionnés dans la vue de gauche sont également mis en évidences sur la vue d'avion centrale et associés aux vidéos de la partie droite 33

2.2.4. Visualisations graphiques

Comme le montrent les exemples des analyses pratiquées à Lyon/Saint-Étienne et Utrecht, les visualisations graphiques permettent aux chercheurs d'avoir des visions différentes des données, et aident souvent à isoler des phénomènes intéressants, ou fournissent une vision intuitive de ce qui s'est passé. Une forme typique de visualisation graphique est la représentation symbolique des évènements sur un axe temporel horizontal. Ce type de représentation est bien ancré dans les habitudes, et se retrouve dans SAW (Synchronized Analysis Workspace, Goodman *et al.*, 2006), ABSTRACT (cf. Figure 38), Replayer, DRS et d'autres encore. Cette représentation permet d'explorer la dimension temporelle des données. Suthers *et al.* (2008) ajoutent à ces représentations des annotations complexes, enrichissant la visualisation au moyen des connaissances acquises au cours de l'analyse, et permettant de communiquer cette connaissance à une audience plus vaste. Teplovs *et al.* (2007) utilisent les données des fichiers de traces d'un forum structuré pour générer une visualisation interactive des messages aux propriété similaires (même auteur, même fil de discussion, proximité sémantique). Il existe de nombreux types de visualisations dont le but est de représenter les connaissances analytiques à propos de données, telles que les représentations de conceptions sous formes de diagrammes argumentatifs (Prudhomme *et al.*, 2007) ou encore les visualisations liées aux outils d'awareness (van Diggelen *et al.*, 2008).

[33]. http://www.dcs.gla.ac.uk/%7Emorrisaj/Replayer.html

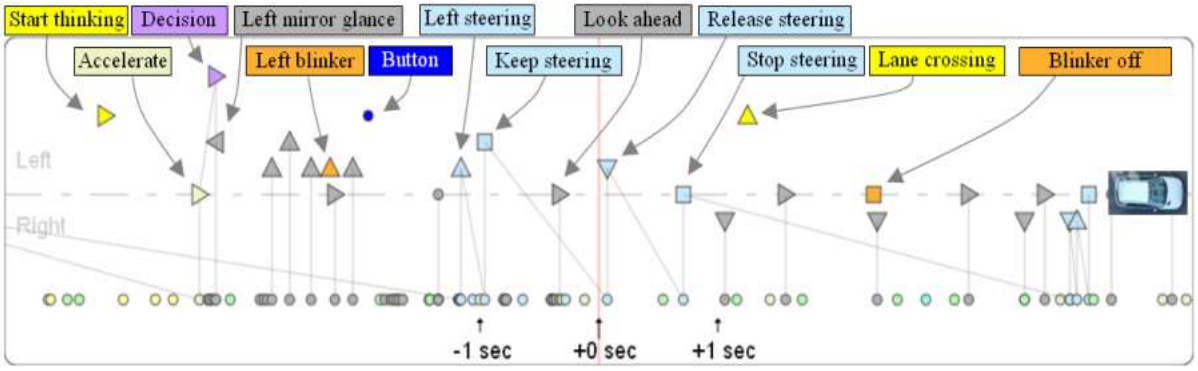


Figure 38. Représentation graphique symbolique des évènements enregistrés lors d'un changement de file sur une autoroute, visualisés avec ABSTRACT (http://liris.cnrs.fr/abstract/).

2.2.5. Echanges et Interopérabilité

Reffay *et al.* (2008) font remarquer l'importance de pouvoir échanger des corpus dans le domaine du CSCL. Kahrimanis *et al.* (2006) examinent comment une meilleure interopérabilité entre les traces produites par les outils de collaboration médiatisée par ordinateur et les outils d'analyse peut être obtenue. Nos études de cas nous montrent que les chercheurs ont des pratiques éprouvées, qui sont confortées par l'usage, et qu'ils hésitent à s'en remettre à un outil unique. Nous nous intéressons nous-mêmes au fait de pouvoir partager des analyses à des fins de validations (telles que la méthode des juges), ou pour acquérir une compréhension croisée en combinant les analyses d'experts de différents domaines (Prudhomme *et al.*, 2007).

2.2. Résumé

Dans les exemples ci-dessus, nous avons mis en évidence la variété des artéfacts que créent les chercheurs souhaitant analyser un *processus* d'interaction, ainsi que les manipulations que les chercheurs effectuent sur ces artéfacts. Dans la section suivante, nous regarderons comment modéliser ses artéfacts et manipulations en vue de la création d'un environnement informatique qui assiste ces analyses.

3. Un modèle simple d'analyse

Harrer *et al.*, (2007) ont modélisé le processus d'analyse (cf. Figure 4), dans l'idée de s'appuyer sur ce modèle pour concevoir la structure d'un outil d'analyse et rendre les

représentations interopérables. Ils décrivent les différentes phases de traitement : capture de donnée, segmentation, annotation, analyse, visualisation, interprétation ainsi qu'une boucle d'itération permettant de retourner à l'annotation.

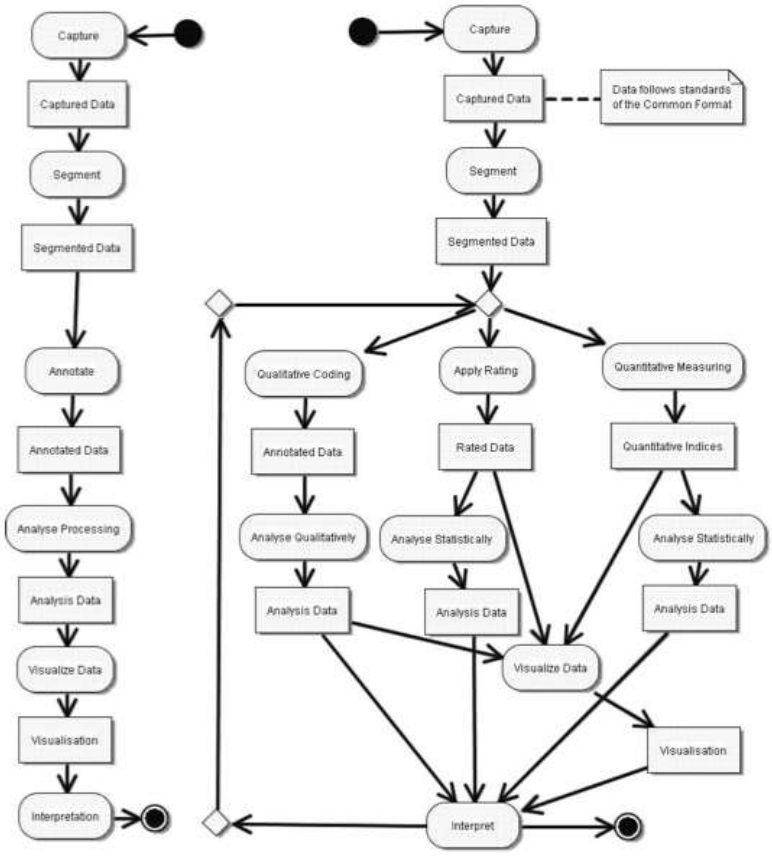


Figure 39. Représentation graphique du modèle de traitement Cavicola (Harrer, et al., 2007, p. 2)

Dans l'état actuel de notre compréhension de l'analyse, basée sur nos études de cas et nos expériences passées (Lund *et al.*, 2007; de Vries *et al.*, 2002; Prudhomme *et al.*, 2007; Baker *et al.*, 2007), ce qui nous semble le plus important dans ce modèle est la boucle : les chercheurs qui en arrivent à la phase d'interprétation et ne sont pas satisfaits des résultats vont affiner leur analyse en réitérant le processus, jusqu'à parvenir à un résultat susceptible d'être transmis à la communauté scientifique. Il ne semble pas non plus nécessaire de parcourir toutes les étapes dans un ordre particulier, ni d'attendre la phase d'interprétation pour décider si une nouvelle itération est nécessaire. Par exemple, la segmentation des données en unités d'analyses est intrinsèquement liée au type de codage auquel le chercheur souhaite procéder ; si ce codage s'avère impossible pour certaines unités, il peut être nécessaire de revoir immédiatement la segmentation sans procéder à la suite de l'analyse.

Nous nous sommes appuyés, pour la conception de Tatiana, sur un modèle similaire (quoique simplifié), qui place plus d'emphase sur la nature itérative de l'analyse (cf. Figure 40). Les analystes évaluent en permanence si la combinaison de leurs données primaires et des artéfacts secondaires est suffisante pour obtenir un résultat. Si ce n'est pas le cas, ils sont amenés à créer un nouvel artéfact, qui permet de mieux comprendre leurs données, ou de réifier leur vision de ces données.

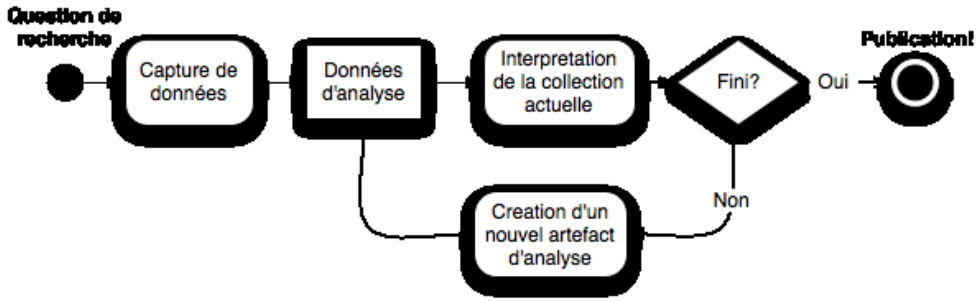


Figure 40. Représentation graphique du modèle de traitement de Tatiana

Dans les schémas d'analyse présentés ci-dessus, nous avons montré la variété des artéfacts que les chercheurs sont amenés à construire. Dans certains cas, cette création peut être automatisée (p.ex. la transformation d'une représentation vers une autre, ou des calculs statistiques). Dans d'autres cas, cette transformation est entièrement manuelle (telle la construction de certaines visualisations), ou peut être aidée par un outil (p.ex. la transcription, l'annotation, le codage). Ces artéfacts sont souvent des représentations des données faisant appel à la dimension temporelle, au travers de différentes approches : la séquence des évènements enregistrés est présentée au serveur par un outil de rejouage (auquel est associée une télécommande permettant la navigation dans les données), ou sous une représentation graphique ou tabulaire, le temps étant associé à l'axe horizontal ou vertical.

Ces artéfacts peuvent être classifiés en trois espèces. Les *assemblages* agrègent les données correspondant à une certaine période de temps, afin de permettre la production d'indicateurs ou de statistiques. Les *analyses* sont des artéfacts qui ajoutent une connaissance créée par le chercheur après examen des données (ces analyses consistent en codages, annotations, liens, etc.). Enfin, nous proposons que les artefacts qui conservent la notion d'ordre des évènements et d'interactions temporellement situées soient nommés *rejouables*. Ce sont des objets qui peuvent être rejoués, synchronisés et analysés. Nous proposons également de considérer le processus d'analyse comme une création itérative de nouveaux artéfacts (tels que les rejouables et les analyses), qui mettent en lumière la compréhension des données par le chercheur, ou lui permettent d'affiner cette compréhension. Nous nous intéressons en particulier aux rejouables, car ils servent fréquemment de sources pour la création de

nouveaux artéfacts, et en particuliers d'autres rejouables. Les transformations de rejouables en autres artéfacts comportent les transcriptions, annotations, codages, visualisations, filtrages, synchronisations, fusions ou assemblages. Il est aussi possible de considérer que la création d'une analyse se ramène à celle d'un rejouable, dans lequel certains évènements sont dotés d'une nouvelle propriété. Dans tous les cas, considérer les analyses comme des entités indépendantes permet de les réifier et de les transmettre à d'autres chercheurs en possession du même corpus, qui peuvent dès lors les intégrer à leurs données.

4. Tatiana : un environnement générique d'analyse

Dans la section précédente, nous avons montré comment certains chercheurs effectuent leurs analyses. Bien qu'il existe nombre d'outils destinés à assister certaines de ces tâches, même les plus généraux de ces outils, comme ActivityLens ou Digital Replay System ont des carences autour d'aspects tels que l'automatisation de transformations, la facilité d'introduire de nouvelles données, ou la possibilité de s'adapter à de nouveaux types d'analyses. Tatiana (Trace Analysis Tool for Interaction ANAlysts) est un environnement conçu pour la manipulation des divers types d'artéfacts décrits ci-dessus, les rejouables en particulier. Nous présentons dans cette sections les diverses caractéristiques du logiciel qui permettent de gérer ces possibilités.

4.1. Tour d'horizon de Tatiana

Le logiciel Tatiana repose sur un certain nombre de concepts et composants fondamentaux (cf. Figure 6) et particulièrement autour de la notion de rejouable. Ces rejouables peuvent être créés automatiquement (par exemple, par importation de données extérieures ou par des transformations) ou manuellement. Une fois créés, les rejouables bénéficient de quatre fonctionnalités : ils peuvent être transformés, analysés, visualisés ou synchronisés.

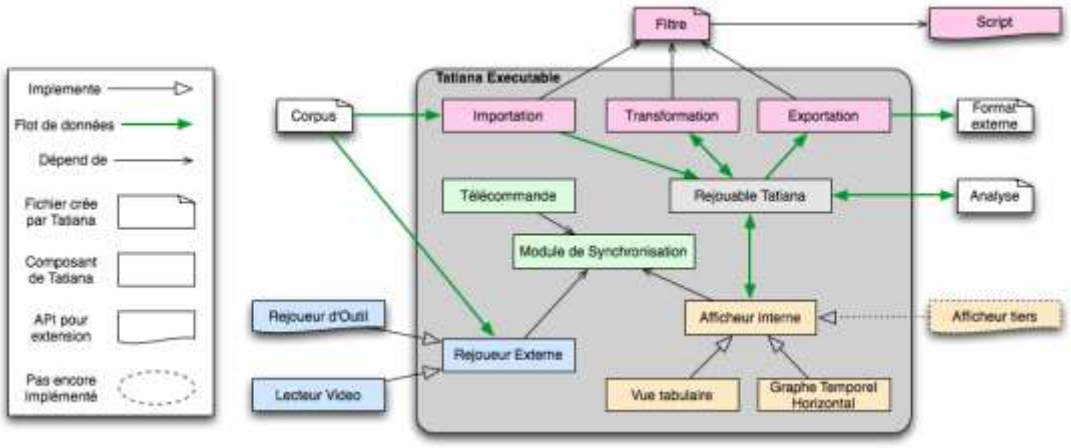


Figure 41. L'architecture de Tatiana faisant apparaître les liens entre les composants, les composants conçus pour être extensibles, et les développements en cours.

4.1.1. Transformations

Les rejouables peuvent être transformés (automatiquement ou interactivement) et exportés. L'importation, la transformation et l'exportation s'effectuent par l'applications de filtres. Un filtre combine des scripts sous forme d'un flux de traitement de données. Les scripts sont de petits programmes (écrits dans le langage XQuery), conçus pour exécuter une opération spécifique : transformer un fichier d'un corpus en la représentation pivot de Tatiana, sélectionner certains évènements d'un rejouables, combiner des rejouables, etc. Comme nous n'attendons pas des chercheurs (qui, pour la plupart, ne connaissent pas XQuery) qu'ils réalisent eux-mêmes leurs scripts, nous avons conçu un éditeur interactif graphique, qui permet de construire un filtre générant le rejouable désiré par le chercheur à partir d'un ensemble de scripts et d'autres filtres existant. Un filtre peut ainsi combiner un script réalisant l'importation de données créées par un logiciel externe et d'autres scripts appliquant divers traitements (regroupement automatique, sélection, extraction, fusion, etc.). Les transformations manuelles interactives permettent d'autres opérations sur les évènements, telles que la destruction, le réordonnancement, le regroupement ou la dissociation en de nouveaux évènements.

4.1.2. Analyse

Dans Tatiana, tous les rejouables peuvent être enrichis par des données d'analyses créées par le chercheur. Deux types d'analyses sont actuellement supportées par le logiciel : les annotations et les catégorisations. La catégorisation consiste simplement à ajouter des

annotations extraites d'une liste fermée de termes, et peut servir au codage, à l'étiquetage ou à la pose de mots-clefs. La liste des catégories disponibles peut en fait être enrichie à n'importe quel moment de l'analyse, ce qui permet de faire évoluer le schéma pour l'adapter aux besoins du chercheur. Ces analyses sont des instances particulières d'un concept générique d'analyse qui permet d'attacher à n'importe quel événement de rejouable une série de données d'analyse supplémentaire.

4.1.3. Visualisation

Tous les rejouables peuvent être visualisés dans différents afficheurs. Il existe actuellement deux types d'afficheurs :

- Une forme tabulaire qui présente les données de manière similaire à celle d'Excel, avec une ligne par évènement et une colonne pour chacune des propriétés associées aux évènements, l'axe des temps étant vertical.
- Une visualisation temporelle horizontale qui est la première version d'un outil générique d'aide à la création automatique de visualisation. Chaque évènement y est présenté sous forme d'un objet graphique dont les propriétés (couleur, forme, taille, position verticale, etc.) peuvent être choisies en fonction des propriétés associées à l'évènement (utilisateur, outil, date, catégories d'analyse, etc.).

Nous envisageons de rendre Tatiana extensible, pour permettre la création de nouvelles formes de visualisation bénéficiant des autres possibilités du noyau du logiciel.

4.1.4. Synchronisation

Enfin, tous les rejouables dans Tatiana peuvent être synchronisés entre eux, et avec des données visualisées dans des afficheurs externes, tels que des lecteurs vidéo ou des outils de rejouage (ce terme de « rejouage » désignant un mode de fonctionnement particulier de certains outils de collaboration, qui peuvent relire les traces qu'ils ont produites lors de leur utilisation, et reproduire sur l'écran le déroulement d'une session antérieure). Tatiana propose un mécanisme permettant de piloter ces outils externes. Le rejouage synchronisé implique que, lorsqu'une date est choisie dans la « télécommande », l'afficheur vidéo et les rejoueurs externes se trouvent immédiatement positionnés à la date correspondante dans leurs média respectifs, les évènements associés étant également mis en évidence dans Tatiana. (cf. Figure 7).

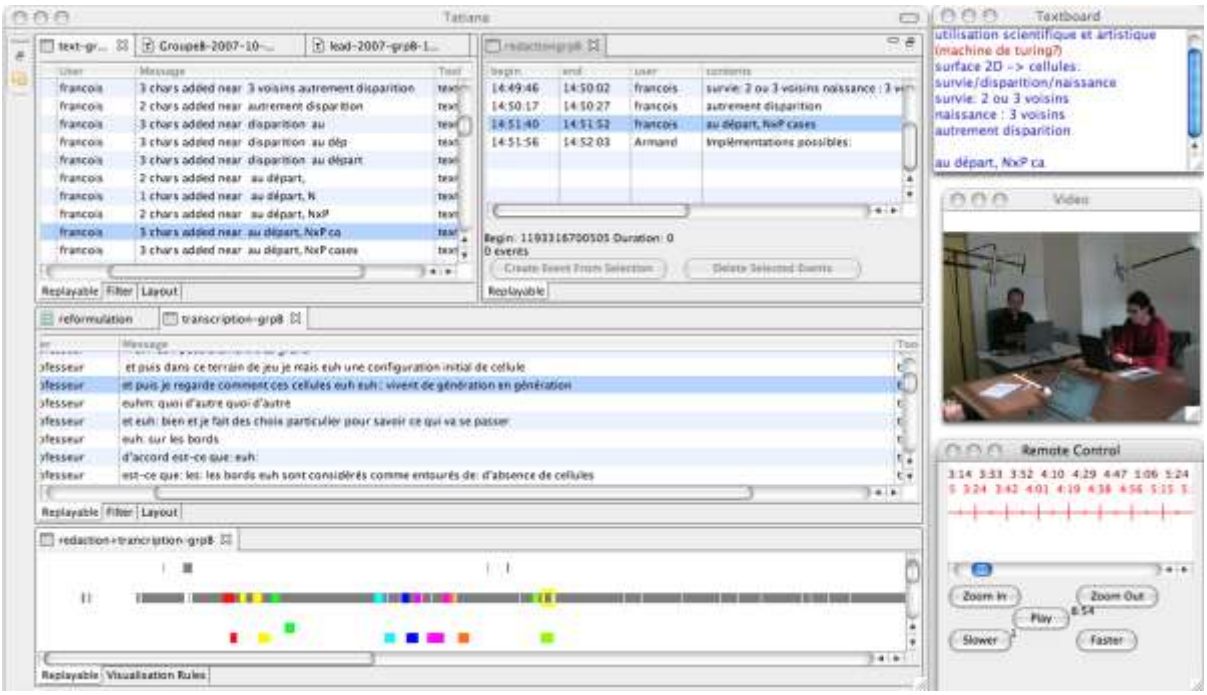


Figure 42. Affichage de différents rejouables dans Tatiana : trace d'un éditeur de texte partagé (haut gauche), transcription des dialogues (milieu gauche), unités de rédaction (haut centre), visualisation des reformulations (bas gauche), ainsi que la « télécommande » (bas, droit) permettant la synchronisation avec des outils extérieurs tels que le rejoueur DREW (haut droit) ou un afficheur vidéo (milieu droit).

De manière générale, la sélection d'un évènement dans un visualiseur de rejouable positionne à cet instant précis tous les autres médias affichés. Par exemple, dans la Figure 7, lorsque le chercheur clique, dans la vue tabulaire, sur l'évènement « *3 chars added near* au départ, NxP ca», le rejoueur montre l'état de l'éditeur de texte partagé juste après avoir inséré ces caractères. La dynamique de l'interaction devient ainsi immédiatement accessible au chercheur, alors qu'elle est difficile à découvrir dans les traces brutes de l'outil. Les opérations de « zoom » permettent d'afficher les épisodes à différents niveaux de granularité. L'existence de ces liens entre les rejouables permet au chercheur de se concentrer sur une visualisation particulière présentant des informations limitées, sachant qu'il lui sera possible, si nécessaire, de se référer instantanément à toutes les autres formes de représentation pour tout complément d'information.

4.2 Exemple d'utilisation de Tatiana

L'usage typique de Tatiana est la création itérative de rejouables et d'analyses, qui permettent au chercheur d'approfondir et de matérialiser sa compréhension du corpus. Ainsi, on peut voir

(cf. Figure 7) comment cet outil a été utilisé (étude de cas Lyon/St-Etienne) pour analyser la reformulation de dialogues en notes prises dans l'éditeur de texte partagé.

Les artéfacts créés peuvent tous être synchronisés au sein de Tatiana. Les transcriptions des dialogues, réalisées au moyen d'un outil externe, sont importées sous la forme d'un rejouable. La traces des interactions médiatisées, de basse granularité, sont regroupées en évènements représentant les unités de rédaction. Les unités de rédaction et les tours de parole de la transcription sont analysés, et reçoivent une même étiquette (ici, une couleur), lorsqu'une reformulation est détectée. Ces deux rejouables sont alors fusionnés, et affichés dans une même visualisation temporelle horizontale, les transcriptions dans la partie supérieure, les unités de rédaction dans la partie inférieure ; l'identité des couleurs montre quelles interventions orales de l'enseignant apparaissent sous forme de reformulation par les étudiants dans l'éditeur de texte. Dans ce travail d'analyse, les transcriptions, regroupements et catégorisations ont été réalisés par le chercheur, aidé par des outils informatiques, mais l'ensemble des autres opérations est automatisé. Même pour ces opérations « manuelles », il devient plus simple d'attribuer un sens et de comprendre le contexte des données produites dans l'éditeur de texte partagé (ici, l'outil DREW a été utilisé : Corbel *et al.*, 2002; Corbel *et al.*, 2003) grâce à l'observation simultanée du rejouage de l'outil et de la vidéo correspondante.

Tatiana n'est en aucun cas spécialisé ou limité à un certain type de données ni à une démarche d'analyse particulière. Tatiana est utilisé pour analyser des données d'interactions synchrones médiatisées par ordinateur, mais aussi en face à face, pour des forums, des blogs et même simplement des vidéos, avec des méthodologies diverses comme l'ethnométhodologie, l'étude de cas et l'analyse quantitative, que ce soit dans une optique hypothético-déductive ou descriptive.

4.3 Usages et limites

Tatiana a été développé en parallèle avec les analyses décrites dans nos études de cas. Ses fonctionnalités nous ont permis de mener à bien notre étude du corpus Lyon/St-Etienne, et sont maintenant appropriées pour la réalisation des autres analyses évoquées. D'autres études vont être réalisées, au travers de l'utilisation conjointe de Tatiana et d'outils extérieurs (tels que Excel ou SPSS pour les analyses statistiques, ou encore Elan pour les transcriptions).

Tatiana n'offre actuellement qu'un support limité des assemblages, ces artéfacts qui représentent le résultat d'un calcul et qui ne sont plus ordonnées dans le temps. Ces objets ne peuvent être synchronisés, analysés et visualisés de la même façon que les rejouables. Cependant, un premier ensemble d'opérations est disponible, utilisant la fonctionalité de transformation de Tatiana, et qui permet la création de certaines statistiques, telles les tables de contingences, qu'il est possible d'exporter vers d'autres formats, comme Excel. Une fois créés, les rejouables et les analyses peuvent être enregistrés et partagés avec d'autres chercheurs ou exportés vers d'autres formats.

5. Conclusions et travaux en cours

Dans cet article, nous avons évoqué nombre des activités effectuées par les chercheurs lorsqu'ils analysent des corpus décrivant des activités de CSCL ou similaires. A partir de ces exemples, nous avons proposé un modèle simple décrivant l'activité d'analyse. Dans ce modèle, les chercheurs créent itérativement de nouveaux artéfacts, qui leur permettent d'améliorer ou de concrétiser leur compréhension du corpus. Nous avons identifié un type particulier d'artéfact, temporellement orienté, que nous appelons rejouable. Nous avons enfin présenté Tatiana, un outil qui permet la création itérative de ces rejouables, et nous avons décrit les principales caractéristiques de ce logiciel en termes de création, exportation, transformation, analyse, visualisation et synchronisation de ces rejouables.

Tatiana répond aux besoins, itératifs par essence, des analyses sociocognitives des interactions, en fournissant un ensemble très souple de transformations et visualisations de données, et en offrant de nombreuses possibilités d'extensions pour répondre aux besoins nouveaux, qu'il s'agisse de créer de nouveaux filtres pour des transformations, ou de créer de nouvelles vues pour l'affichage et la modification de rejouables. Ces visualisations pourraient s'apparenter à celles qui sont proposées dans Suthers *et al.* (2008), ou encore Teplovs *et al.* (2007). L'utilisation de visualisations multiples et synchronisées des données, et la possibilité de faire appel à des afficheurs ou rejoueurs extérieurs permet en partie de pallier les difficultés que peut rencontrer le chercheur qui veut comprendre ce qu'ont pu faire les participants observés. Enfin, grâce à sa possibilité d'enregistrer et de partager des analyses, Tatiana permet aux chercheurs en sciences humaines et sociales de travailler en équipe afin d'intégrer et de comparer leurs analyses.

Nos travaux futurs ont pour objectifs simultanés de mieux comprendre les méthodologies utilisées par les chercheurs en CSCL/CSCW, tout en développant Tatiana en tant qu'environnement d'étude et de gestion de rejouables et autres artéfacts de l'analyse. Nous visons à rendre particulièrement cohérente l'infrastructure du logiciel, et à nous assurer que des chercheurs qui n'ont pas de connaissance particulière en programmation puissent utiliser toute la puissance de cet environnement.

Nous espérons que la diffusion de Tatiana[34] (http://code.google.com/p/tatiana) et d'outils similaires permettra aux chercheurs de tirer un meilleur parti de leurs corpus, tout en partageant ce savoir avec les autres chercheurs. Nous souhaitons également, au travers des usages de Tatiana, acquérir une meilleure connaissance des processus d'analyse sociocognitive des interactions, et rendre plus aisés au sein de la communauté du CSCL l'évaluation et le partage des corpus et des analyses.

[34]. Notons que Tatiana est diffusé gratuitement et sous une licence libre, permettant à chacun de disposer du code source.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 551 I

Gregory DYKE

A MODEL FOR MANAGING AND CAPITALISING ON THE ANALYSES OF TRACES OF ACTIVITY IN COLLABORATIVE INTERACTION

Speciality : Computer Science

Keywords : Trace Analysis, Knowledge Engineering, CSCL, CSCW, TEL

Abstract :

In this dissertation, we address the problem of the socio-cognitive study of human interaction, more particularly in the field of computer-supported collaborative learning (CSCL). The study of such situations can be performed through the analysis of traces (log files, audio, video, etc.) of the activity process, yet presents numerous difficulties:

- Large quantity of complex multimodal data from a variety of sources.
- Data which is strongly dependent on the context in which it has been produced.
- Variety of possible methodological and epistemological approaches to analysis.
- Difficulty in sharing and re-using data and analyses performed on this data.

Our work has been on the reduction of these difficulties and, in this dissertation, we present our three main results. On one hand, we propose a description of the process of analysis of such data, as well as a generic artefact which covers a large number of the analytic artefacts we have observed and which we call a replayable. On the other hand, we present a study and a modelling of replayables, and describe the four fundamental operations which can be applied to them: synchronisation, visualisation, transformation and enrichment. Finally, we describe the implementation of this model in an environment that assists analysis through the manipulation of replayables, which we evaluate in real-life research situations.

Tatiana (Trace Analysis Tool for Interaction Analysts – http://code.google.com/p/tatiana), the resulting software environment, is based on these four fundamental operations and integrates numerous possibilities for extending these operations to adapt to new kinds of analysis while staying within the analytic framework afforded by replayables. This tool is used by over 10 research teams (France, United Kingdom, the Netherlands, Denmark, Hong-Kong) to study a variety of phenomena.

This research theme addresses the wider issues of scaling up the analysis of interaction data and sharing corpora and analyses of these corpora to allow validation, replication and collaboration among analysts. The object we have defined opens up the applicative domain of trace analysis to many computer science disciplines (information retrieval, data mining, knowledge engineering, HCI, artificial intelligence), while simultaneously taking into account the particularity of these kinds of analyses and the artefacts which are used to perform them.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : 551 I

Gregory DYKE

UN MODÈLE POUR LA GESTION ET LA CAPITALISATION D'ANALYSES DE TRACES D'ACTIVITÉS EN INTERACTION COLLABORATIVE

Spécialité : Informatique

Mots clefs : Analyse de Trace, Ingeniérie des Connaissances, CSCL, CSCW, EIAH

Résumé :

Dans cette thèse, nous nous adressons au problème de l'étude socio-cognitive d'interactions humaines, plus particulièrement dans le domaine de l'apprentissage collaboratif médiatisé par ordinateur. L'étude de telles situations passe par l'analyse de traces (fichiers de log, audio-vidéo, etc.) du processus d'activité et présente de nombreuses difficultés :

- Grande quantité de données multimodales complexes issues de sources différentes.
- Données fortement dépendantes du contexte dans lequel elles ont été produites.
- Variété des approches méthodologiques et épistémologiques d'analyse possibles.
- Difficulté de partage et réutilisation de données et d'analyses effectuées à partir de ces données.

Notre travail a porté sur une réduction de ces difficultés et nous présentons dans cette thèse nos trois résultats principaux. D'une part, nous proposons une description du processus d'analyse de ce genre données ainsi qu'un artefact générique permettant de recouvrir un grand nombre d'artefacts analytiques que nous avons pu observer et que nous nommons *rejouable*. D'autre part, nous présentons une étude et modélisation informatique des rejouables, et décrivons les quatre opérations fondamentales qui peuvent s'y appliquer : synchronisation, visualisation, transformation et enrichissement. Enfin, nous décrivons l'implémentation de cette modélisation dans un environnement d'aide à l'analyse par manipulation de rejouables que nous évaluons dans des situations de recherche réelles.

Tatiana (*Trace Analysis Tool for Interaction Analysts* – http://code.google.com/p/tatiana), l'environnement logiciel résultant, est basé sur ces quatre opérations fondamentales tout en intégrant de nombreuses possibilités d'extension de ces opérations pour s'adapter à de nouvelles formes d'analyse sans sortir du cadre de l'analyse au travers de rejouables. Cet outil est utilisé par plus de 10 équipes de chercheurs (France, Royaume Uni, Pays-Bas, Danemark, Hong Kong) sur des thématiques et objets d'étude variés.

Cette thématique de recherche s'introduit dans une problématique plus large de passage à l'échelle sur l'analyse de données d'interaction, de partage de corpus de données et d'analyses collaboratives et incrémentales sur ces corpus. L'objet que nous avons défini et implémenté permet à de nombreuses disciplines informatiques (recherche d'information, fouille de données, ingénierie des connaissances, interaction homme-machine, intelligence artificielle) de trouver un nouveau champ d'application dans l'analyse de traces, tout en tenant compte de la particularité de ces objets d'étude et des artefacts qui en sont issus.