Trace Analysis in Instrumented Learning Groupware: an experiment in a practical class at the university

Stéphane Talbot, Christophe Courtin

Trace Analysis in Instrumented Learning Groupware: an experiment in a practical class at the university

Stéphane Talbot, Christophe Courtin
Laboratory Systèmes Communicants – University of Savoie
73376 Le Bourget-du-Lac Cedex
France
{Christophe.Courtin,Stephane.Talbot}@univ-savoie.fr

ABSTRACT
Software for supporting students in learning in a collaborative way is very often less productive than expected. We have defined models to collect and analyse traces of learning activities in instrumented collective learning situations (ICLS). We have conducted an experiment, to test a prototype of an observation station, with students and a teacher during a foreign language course at the university. We use this prototype to point out how software tool use models can provide a human observer with information to analyse collaborative learning activities..

KEY WORDS
trace, observation, intrumentation, awareness, collaborative learning

1. Introduction

E-learning technologies are becoming increasingly popular, but it is often difficult to take advantage of collaboration in the learning process. Indeed, collaborative learning systems should enable participants to observe group activity, in order to be aware of the work in progress. An observation station should give as much information as needed for adapting activity in an instrumented collective learning situation (ICLS). As far as the teacher is concerned, such information may lead her/him to modify the pedagogical scenario over time [1], for instance by adding exercises or specific explanations in the event of a failure. Observation of students' work may allow the teacher to detect effective strategies. For students, this observation is useful to situate their own work in the group, to get information about their own errors, etc.

In this paper, we present an experiment which highlights the feasibility of an observation station, from the technical results of a prototype. This first experiment is a starting point to evaluate the benefits of such a system in collaborative learning activities

2. Objectives

With this experiment, we aim to test our intra and inter use models of software tools [2]. These models, which describe the expected use of each software tool of an ICLS and between tools, are represented by a set of rules. A use model has to facilitate activity interpretation by increasing the abstraction level. An abstraction level corresponds to an observer's specific point of view (e.g. pedagogical, communicational, etc.). A use model is based on structured information (collect model [3]) with specific semantics, that we call "templates".

3. Experiment

Figure 1 : experiment

Our experiment was carried out during a practical class in an English course (foreign language) with students at the university. The work consisted in teaching English vocabulary through the study of a text in English, to students working in pairs, and placed on separate workstations, but communicating with an appropriate software tool ("coffee-room" which is a structured chat room). The students were free to organize their work in pairs, but each member had to participate actively in the exercise. This consisted in defining a set of situated English words. The work was finished when all the definitions had been completed, or when the class was over.

The main final objectives defined by the teacher were to promote knowledge sharing in pairs, and to detect one's own errors.

In the experiment, the production tool is called "jibiki" (an asynchronous collaborative editor) and the communication tool is called "coffee-room" (a chat room in which communication spaces are represented by tables).

The experiment trace technique is based on the instrumentation of the software tools which are used in an instrumented collective learning situation (ICLS).

The instrumentation technique is equivalent to the log system one [4], except for the fact that it takes place at the level of the software tools themselves. We observed promising results with the log system [5], and we therefore propose a technique as an extension of it. Both techniques have advantages which compensate for their respective drawbacks, and we plan to use both of them in our architecture. Indeed, we will present the possibility of considering other trace sources in our analyses [6]. This flexibility is possible because collect and trace analysis modules are separated in our system architecture.

As instrumentation takes place at the software tool level in an ICLS, it enables one to provide traces with an abstraction level close to that of the human observer, that is to say one which enables the description of actions according to a software tool use model, defined by the observer her/himself.

The log systems, which represent low level data on a server, enable one to collect all the events generated by the system when using the software tools of the ICLS. However, this technique generates a great deal of noise [7], and it is difficult to interpret low level events. Furthermore, we maintain that the system is unable to provide all the elements useful for the description of actions (we will see an example hereafter).

The instrumentation technique is based on the idea of the nature of the traces we wish to obtain. Thus, we define a use model for the software tools of the ICLS, which describes all the actions expected in this kind of activity. After having defined the corresponding observables, instrumentation consists in sending explicitly associated information, called signals, from the software tools themselves.

The use model of software tools in ICLS is represented by a system of rules in order to recognize potential actions from signals and sequences [2]. The rules, triggered by signals matching their conditions, provide higher abstraction level traces, called level one sequences. By extrapolating this mechanism, level n sequences may be created from signals and previously created level n-1 sequences.

Whatever the technique used, we observe that some data cannot be provided by the system itself when actions start. As an example, communication between two given participants about a given topic. This meaningful information is not provided by a single event-like "open communication tool". Therefore, our assisted-analysis system generates explicitly high abstraction level traces with a low granularity structure (signals). Working on these signals, which are reinjected into those produced by the system, is possible in asynchronous mode for back-office assisted analysis. In this case, we identify the analyser as being the source of the signal (as for the sequences), and not the system itself.

In short, the instrumentation technique enables one to enrich the traces produced by the system, with signals and sequences.

4. Trace format

With the trace manager, we are able to manage two kinds of activity traces :

- signals which correspond to time pinpoints and elementary elements (e.g. a user action, a state modification of the system, and so on);
- sequences which split into a chronological succession of signals or sub-sequences. Obviously a sequence also has some duration and normally should make sense to understand what happened with the tools.

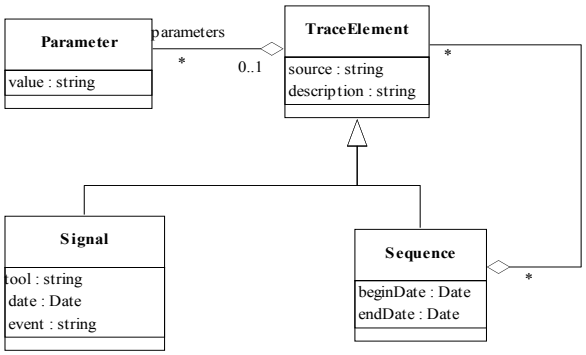

Figure 2 : UML model for traces

To prepare the experiment we have instrumented the two tools used for this one: the Coffee Room and the Jibiki. So each single significant action of either the students or the teacher were converted into signals and send to the trace manager.

To achieve this goal we have modified the Coffee Room (the tool the students use to chat to each other). Now it generate a signal at user connection or disconnection, table creation or destruction and when one user sends a message at some table.

These signal contains :

- a source (the people or the tool which has generated the signal – in our experiments it's always the Coffee Room or the Jbiki);
- a tool (the tool or the instance tool in which the event has taken place – in our experiments its always an instance of Coffee Room or the Jbiki);
- a date (the timestamp which says when the event happened);
- an event id. The list of possible events will of course depend on the tools we use: connection, disconnection, message emission, for the Coffee Room);
- a textual description and
- a list of parameters (which contains the variable parts of signals). In this list we should find everything that is needed to understand what has happened. For example who and what is involved in the event. Obviously, this part also will change with events and tools.

The same action have been conducted for the Jibiki. So each action a user can do (user login, start or stop the edition of an entry, changing its state: from editing to reviewed, finished or validated) a signal is emitted and collected by the trace manager. All theses signals are also described by means of pertinent parameters.

The sequences are more complex than signals. So we will rather obtain them from the analyzer part of the observation station: that is its job ton interpret the signals into meaningful sequences suitable to understand the activity of the experiment participants. Each sequence is composed of signals or subsequences and, as signals, have parameters.

For example if somebody as spoken at a table, the other people who were at the same table would have heard him/her, according to that one could want to record that "communication act" and all the details : a sequence can do that.

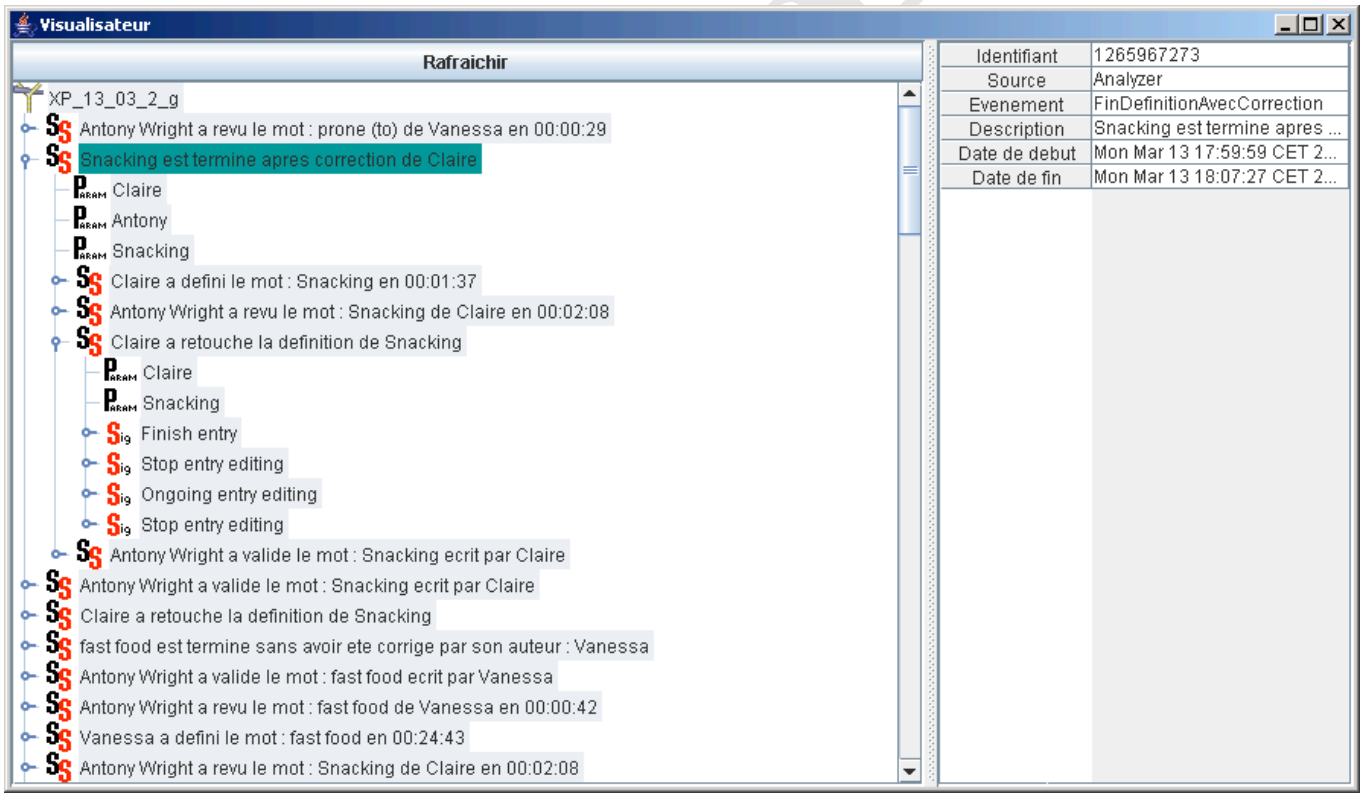

Figure 2: visualization of sequences

For example when somebody chats at one table, the Coffee Room sends a chat signal to the trace manager. This signal has three parameters : The first is the name of the people who has talk, the second is table identifier and the last one is the emitted message.

In the same way, if participants chats a lots than we will surely have different "communication acts" betweens the same participants and perhaps have some interest to interpret all these "communications acts" as a "conversation". A sequence can also be used to group different sub-sequences together and give a new interpretation for this group of sequences.

In our implementation, each sequence is stored with:

- a start date (timestamp – the beginning of the sequence);
- a end date (timestamp – the end of the sequence);
- a type id, which characterize the kind of sequence;
- a source (the people or the tool which has recognized the sequence – in our experiments it's always the analyzer);
- a textual description and
- a list of parameters (which gives all significant details of the memorized episode).

During the experiment we have especially tried to identify sequences which one could associate with cooperation activities (communication acts, conversation, …) or with the progression of the tasks assigned to the students or the teacher (definition proposal, evaluation, correction and validation, and so on).

As stated previously, actually the trace manager needs to be associated with a trace analyzer in order recognized sequences. The analyzer is able to identify the sequences in line (during the experiment) or off line (when the experiment is over).

The sequences the analyzer should recognized are described with rules. Each subpart of the sequence fits with a pattern. So a rule a composed of different patterns which can match against signals or sub-sequences (we can put variables in the patterns). Using these rules the analyzer search the signals, then the sequences which match the patterns and store the new recognized sequences inside the trace base.

We have actually four operators that can be used to group patterns inside rules : **and**, **or**, **negation** and **next**. The three first ones have their standard logical interpretation when the last one is used to specify that sub-sequences or signals should be sequentially ordered. Moreover, in order to avoid problems associated with the use of negation (the classical non monotony problem due to negation in rule based systems), the negation operator has always to be used in association with next.

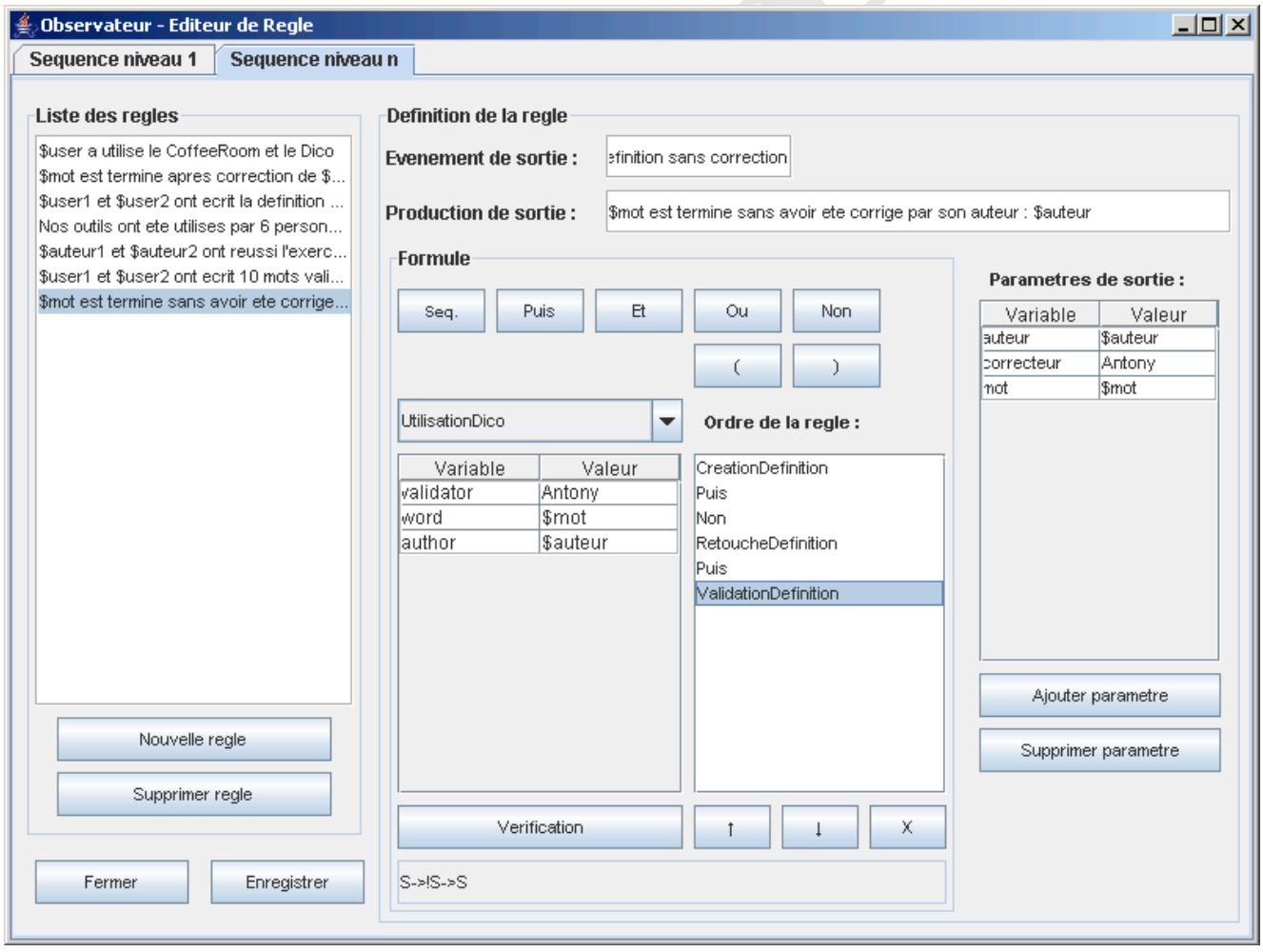

Figure 3: the rule editor

3. Conclusion and perspectives

The experiment presented in this paper is part of an overall project about the development of an observation station. The prototype we have developped for this experiment allows us to reach our main objectives in terms of trace analysis. This first step contributes to the validation of our model analysis based on rules. As our system is open source and based on programming standards, we plan to introduce other trace elements (e.g. from agents [5] or database [7]) in the observation station, to be run in the analysis system. Such an operation would need restructuration of collected information to match with our collect model [3].

The next step consists in evaluating final users' objectives on the pedagogical level. An other experiment, which implicates social and pedagogical experts, will be led in a near future. In order to reach these objectives, the future prototype needs to take into account signals generated by the analysis system itself, because of the system limits to produce some information (e.g. predictive actions with high abstraction level).

References

[1] S. Talbot, P. Pernelle, "Helping in collaborative activity regulation: modeling regulation scenarii", 15th French-speaking conference on human-computer interaction (IHM 2003), T. Baudel, Ed. IHM 2003, vol. 51. ACM Press, Caen (France), November 25-28, 2003, pp. 158-165.

[2] C. Courtin, and S. Talbot, "Trace Analysis in Instrumented Collaborative Learning Environments", 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (The Netherlands), July 5-7, 2006, pp. 1036-1038.

[3] C. Courtin, and S. Talbot, "An Architecture To Record Traces In Instrumented Collaborative Learning Environments", International Conference on Cognition and Exploratory Learning in Digital Age (CELDA'05), IADIS, Porto (Portugal), December 14-16, 2005, pp. 301-308.

[4] J.-M. Heraud, J.-C. Marty, L. France, T. Carron, "Helping the Interpretation of Web Logs: Application to Learning Scenario Improvement.", Workshop Usage Analysis in Learning Systems, 12th International Conference on Artificial Intelligence in Education (AIED 2005), Amsterdam, The Netherlands, July 18th, 2005.

[5] T. Carron, J.-C. Marty, J.-M. Heraud and L. France , "Preparing An Observed Pedagogical Experiment", International Conference on Cognition and Exploratory Learning in Digital Age (CELDA'05), IADIS, Porto (Portugal), December 14-16, 2005, pp. 526-531.

[6] S. Metz, I. Boukhriss "La conception du campus numérique VCIEL : compromis pour le maintien d'identités, Innovation, Formation et Recherche en Pédagogie Universitaire", XXIIIème Congrès de l'Association Internationale de Pédagogie Universitaire, Monastir, (Tunisie), May 15-18, 2006.

[7] R. Smith, and B. Korel, "Slicing Event Traces of Large Software Systems", poster in proc. of 4th International Workshop on Automated Debugging (AADEBUG), Mireille Ducassé (ed), Munich (Germany), August 28-30, 2000.

[8] L. France, J.-M. Heraud, J.-C. Marty, T. Carron, "Help through visualization to compare learners' activities to recommended learning scenarios", 5th IEEE International Conference on Advanced Learning Technologies (ICALT 2005), Kaohsiung (Taiwan), July 5-8, 2005, pp. 476-480.