Tracks Analysis in Learning Systems: A Prescriptive Approach

Sébastien Iksal
Computer Sciences Laboratory
University of Le Mans / IUT de Laval
Laval, France

Abstract

Nowadays, numerous teachers are using Technology Enhanced Learning systems in various domains. The tracks analysis concerning students' actions logged by Technology Enhanced Learning systems can help teachers to improve their pedagogical scenarios making them more relevant to students. Very often analysis solutions are developed ad hoc and without considering the teachers observational needs. In our work, we consider i) that it is very important to identify teachers' needs; ii) to provide a formal description of indicators for capitalization; and iii) to facilitate sharing and reusing of indicators. In this paper, we present our proposal for obtaining these goals. Firstly, we describe our approach for tracking analysis, and the declarative language including the formalization of the calculation method for pedagogical indicators. Then, we present our developed tools (editor and interpreter). We also briefly present an experiment which validates our proposal.

1. Introduction

Nowadays, numerous Technology Enhanced Learning systems are available for teachers. Most of these systems need some kind of feedback on the usage in order to improve them. The software development process should explicitly integrate a usage analysis phase, which can provide designers with significant information on their systems' uses for a re-engineering purpose [1]. Automatic usage analysis is often made by mathematicians or computer engineers. In order to facilitate the appropriation, the comprehension and the interpretation of results by instructional designers, who are the main actors of an e-learning system development process, we think they should be fully integrated in this analysis phase. In TEL systems, teachers have two different roles: instructional designer and tutor.

The research contribution we present in this paper is fully in line with our approach to the engineering and re-engineering of TEL systems, where we particularly stress the need for a formal description of the design view, in terms of scenarios and learning resources, to help the analysis of observed uses (i.e., descriptive scenarios) and to compare them with the designer's intention (i.e., predictive scenario) [2][3], in order to enhance the quality of the learning. A set of observation needs are implicitly defined when designers use an Educational Modeling Language (EML) such as Learning Design [4] to explicit their pedagogical scenario. But there is no real interaction in order to make more explicit the observation needs. Thus, one of the student data analysis difficulties resides in the correlation between these needs and the tracking means provided by the educational environment (not only the computer-based system, but also the whole learning organization, including humans and data collection vectors such as video recorders, questionnaires, etc.).

Our aim is to provide the actors of a TEL System with a language dedicated to the description of the tracks and their semantics, including the definition of the observation needs and the means required for data acquisition. They usually denote a significant factor event that happened during the learning session, on which users (designers, tutors, learners, and analysts) could base some conclusions concerning the quality of the learning, the interaction or the learning environment itself.

It is very important to consider that we are not working directly in the field of data mining as it is considered by [5]. We focus on the description by the teacher of his/her observational needs, the automatic evaluation of indicators and their capitalization. It is possible, in our mind, to integrate data-mining algorithms as a mean to evaluate an indicator.

In this paper, we will present our prescriptive approach and the Usage Tracking Language (UTL) which is our proposal for a formal description of indicators. The interpreter and the data editor will be introduced. Next, we will propose some use case for the language and in the last part, we will give a description of an experiment on UTL. We will conclude with some future works.

2. The prescriptive approach

In our approach (Figure 1), we consider that the teacher is the most suited to identify what is interesting to observe during the learning session. So, we focus our analysis on teacher observational needs which are given as observational instructions: what is interesting to observe, why it is important (assessment, regulation, adaptation, re-engineering, etc.), and what are the values needed to understand the situation.

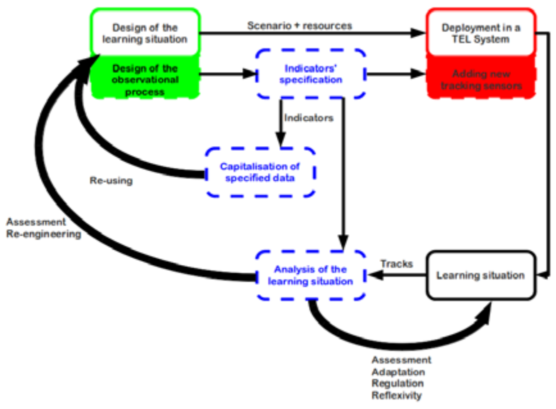

Design of the learning situation
Scenario + resources
Deployment in a TEL System
Design of the observational process
Indicators' specification
Adding new tracking sensors
Indicators
Capitalisation of specified data
Re-using
Assessment Re-engineering
Analysis of the learning situation
Tracks
Learning situation
Assessment Adaptation Regulation Reflexivity

Figure 1. Observation by prescription life cycle

These instructions are used to specify indicators by means of tracks available in the TEL system. If it is not able to provide information needed by the indicator, we have to consider the integration of (1) new sensors directly in the TEL system or (2) a new actor in the learning session to observe and relate the facts.

During the learning session, indicators are evaluated by the analysis process and values are given to the teacher for re-engineering or assessment purpose, to the tutor for regulation, adaptation during the session or assessment, and to students in order to better understand their learning (reflexivity).

The indicators' specifications are capitalized, because we consider that the teaching community has to be able to share and reuse indicators in their activity. That is why the specification has to be independent from the track format and the educational modeling language.

3. Usage Tracking Language: UTL

By considering the observation process on the entire life-cycle of a pedagogical scenario, we have identified three main roles. The teacher (or instructional designer) describes the scenario and specify his/her observational needs. It is very important, at this step, to keep the teacher free to develop his/her ideas. That is why, we are not interested in providing a specific description language. Next, the analyst has the responsibility to transform the observational needs into indicators. He/she has to negotiate with the developer which is able to identify all possibilities of tracking inside the learning environment, and also decides if it is possible to add new ones. The analyst is the link between the pedagogical domain and the technical one. This proposal focuses on a modeling language for the analyst. Concerning the observational need specification by a teacher, we collaborate with the PROTON project of our lab [6].

The Usage Tracking Language (UTL) aims to be neutral regarding technologies and EMLs, and has to be instantiated on the EML used to define the pedagogical scenario and the tracks formats. Moreover, this language allows the structuring of tracks, from raw data – those acquired and provided by the educational environment during the learning session – to indicators [7] which mean something significant for its user. Very often, indicators are build ad hoc for a specific scenario and an specific learning environment. DIAS [8] proposed fifty-two indicators. Synergo [9] makes some statistics (the number of messages exchanged by each learner in a chat session, the number of objects carried out by each actor, etc). Some projects such as TBS (Track Based Systems [10]) are working on a description of tracks transformations in order to obtain an indicator after numerous transformations, but they are not really generic and they don't take into account the teachers' needs.

We describe now the conceptual model and the data combination language which allow the automatic calculation of indicators as well as their capitalization.

3.1. The Conceptual Model

The Usage Tracking Language is composed of three parts (cf. Figure 2). The UTL Scenario dataset (UTL/S) is the link between the pedagogical scenario and indicators. We describe the tracking purpose of each indicator with concepts of the scenario representation model. The UTL Tracks dataset (UTL/T) describes the specific track format of the learning environment, it is used to identify tracks and extract their content in order to convert them in a UTL syntax. Finally, the UTL Pattern dataset (UTL/P), which is the core of UTL, contains the representation of all data needed to valuate indicators. That specific part of UTL is independent of the scenario representation model and the track format.

European projects have influenced our proposal ([11] [7] [12] [13]) on the UTL/P dataset. We have identified two main data types for tracks: the *derived-datum* type and the *primary-datum* type. The primary data are not calculated or elaborated with the help of other data or knowledge. They could be recorded before, during or after the learning session by the learning environment, such as, for instance, a log file, a video tape of the learner during the session, a questionnaire acquired before or after the session, or the sets of posts in a forum, etc. This kind of data is classified as *raw-datum*. The *content-datum* type concerns the outcomes provided by the learning session actors (learners, tutors and/or teachers). These data are mainly the productions of the learners, intended to be assessed, but they also could be, for instance, a tutor report on the activity of a learner, or on the use of a resource. Both of these data have to be identified in the collection of tracks provided by the learning environment, in terms of location and format. The *additional-datum* type qualifies, as DPULS did, a datum which is linked to the learning situation and could be involved in the usage analysis. The derived data are calculated or inferred from primary data or other derived data.

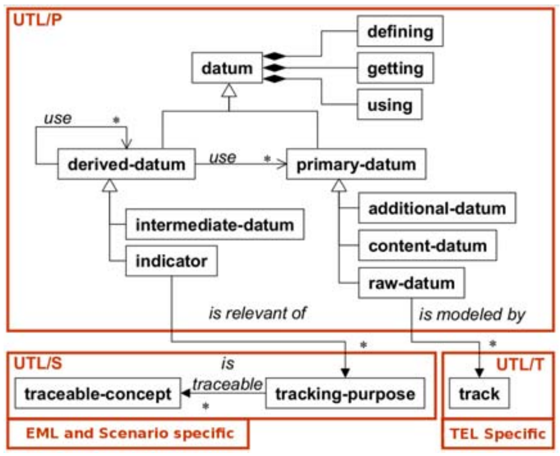


Figure 2. UTL conceptual model

The *indicator* type qualifies derived data which have a pedagogical significance. Thus, an indicator is always relevant to a pedagogical context: it is always defined for, at least, one exploitation purpose, and linked to, at least, one concept of the scenario. We will detail this specific aspect further in the paper. A derived datum which has to be calculated but which has no pedagogical significance is qualified as an *intermediate-datum*.

Each data of the UTL/P dataset is described according to three facets (cf. Figure 3):

- The Defining (D) facet models the tracks needs;
- The Getting (G) facet models the tracks means;
- The Using (U) facet models the tracks uses.

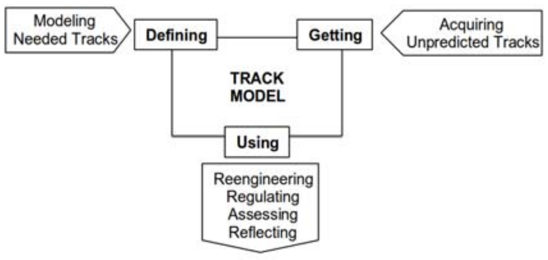


Figure 3. The DGU model

This DGU model allows two processes for modeling a datum: the predicted one, when the designers, during the design phase, declare the datum as needed, and the unpredicted one, when the datum is collected or calculated without an explicit designer's request. In the first process, the Defining and the Using facets are filled first; then the Getting facet is discussed with developers. This is the way one could provide, for instance, examples and descriptions, rather than a specific technique or tool. In the second process, developers and/or analysts fill the Getting facet first, and then the Using and Defining facets are discussed with designers.

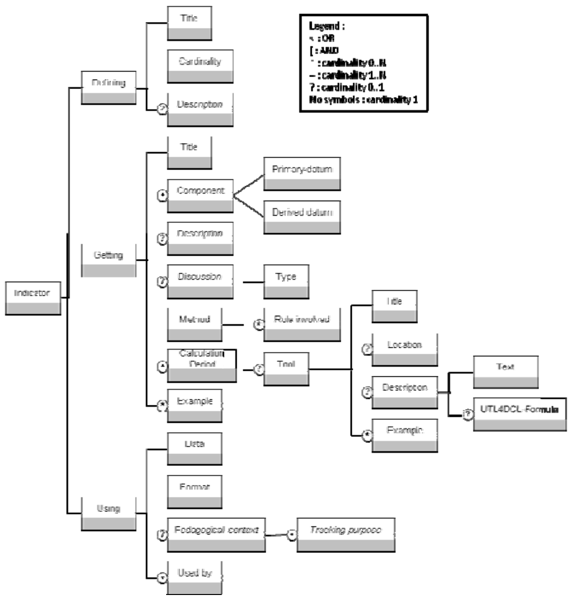


Figure 4. Information model of an indicator

For instance, in the indicator information model (cf. Figure 4), we have its name, a textual description of the indicator and the cardinality (is it a single value or a set of values such as a progression?) in the *Defining* section. In the *Getting* section, the list of data needed to evaluate the indicator is given. It is composed also of the method to compute it, we specify if the method is manual, automatic or semi-automatic, roles involved in the method if needed, the formula or the function used. UTL can be used to produce indicators during the session, so it is also possible to define when it is necessary to evaluate it (at a specific time from the beginning of the session, with a specific user action, etc.). We integrated a *discussion* field for capitalization purpose. It is interesting for the community to evaluate the indicator, in order to improve the re-use. The *Using* section is composed of the *Data* field which contains the evaluation, the *format* field to describe the representation form of the evaluation, and also a *used-by* field to manage a graph of relationship between UTL data. The indicator is different from the other data because it corresponds to a particular observational need.

Each indicator is associated to a tracking purpose, that is to say one or more concepts of the pedagogical scenario, and qualified with one of these categories: assessment, regulation, re-engineering and reflecting. We qualified the indicator with the usage considered by the teacher.

The other datum with particular fields is the raw-datum. It is linked to the track format through the UTL *Track* element. We have considered that it is necessary to specify how the track is composed in order to associate semantic to all components. That specific section of the raw-datum describes the way of storage for tracks (text file, XML file, Database), and also the way of extracting information (query, string composition, etc. This specific UTL Track element has to be modified in order to follow the track representation format.

Table 1. Instance of a DCL4UTL formula

```
cal {
  $CDtmp=end(CD.* as sort(CD.time_receive));
  // Variables initialization
  ...
  if (e=="SQ") {cq=0;}
  else {
    $t3=index(R.E[T] as filter(R.user==u and
R.session==s and R.E[T]&lt;=t1) sort(R.E[T]),
numSQ) ;

    cq=R.E.NQ as filter (R.user==u and
R.session==s and R.E[T]==t3);}

    $pro=CDtmp.datum.projet[nom];
    $nameClss="Point"; $cc="1";
    $cnt=
count(CDtmp.datum.projet.classe[nom==nameClss]);

    $vl=""; $st="PAS_OK";
    if (cnt &gt;0) {vl=nameClss; st="OK";}
    else {vl="";}
    $cl="OK";
    $rf="Point";
    $lo=concat("Projet ", pro);

I.result[session=s][student=u][time=t1][event=e]
[value=vl][status=st][calculable=cl][reference=r
f][location=lo][currentQuestion=cq][concernedQue
stion=cc][category="specifique"][criticalLevel="
"]= "";
  }
where
I=PEDALO-I-ClassPointExist.using.data.indicator,
CD=PEDALO-CD-
ClassDetail.using.data.contentDatum,
R=PEDALO-RD-
SelectionQuestion.using.data.rawDatum
```

3.2. The data combination language: DCL4UTL

We consider that UTL data patterns have to be executed. For these reasons, we have proposed an extended part for UTL called Data Combination Language for UTL (DCL4UTL) [14]. This language is designed to be generic, inspired from classical query languages and integrated in UTL (Table 1). Because of UTL is independent from the system and the structure of tracks storage, the language proposed has to be also independent from the storage. It has to be fully in line with UTL information models. Its focus is to combine the raw data and/or other UTL data in order to establish a new datum and to capitalize combination methods for their re-usability. In DCL4UTL, each calculation is also a transformation from a track model to another.

While UTL allows capitalizing tracks in the form of data design patterns, DCL4UTL completes these data patterns by adding calculation methods for making them executable.

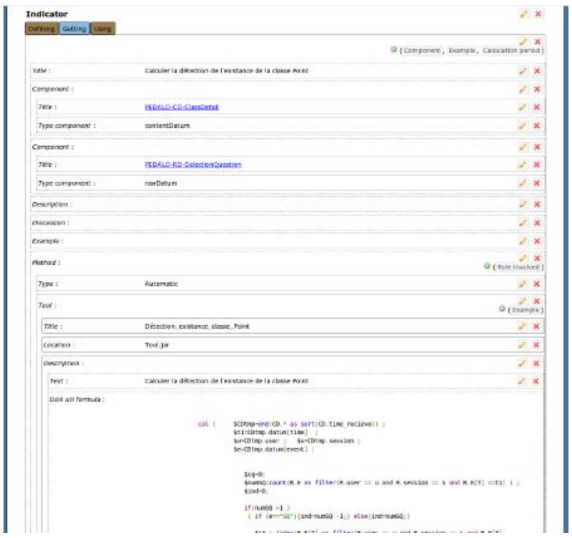
The combination of UTL and DCL4UTL allows not only modeling tracks but also producing indicators. They are addressed to human as well as to the machine. However, both UTL and DCL4UTL are used by data analysts; teachers are not the main users of these languages. Teachers are concerned by the use of the indicators' calculation results. DCL4UTL can be used to calculate both quantitative and qualitative indicators. For example, the average time each student spends on each question, or for verifying if a student remakes the question "A" while performing the question "B", etc. It operates on the *Format* and *Data* field of the UTL Using facet. The *Format* field defines how instances of data (in the UTL meaning) are represented once they are generated from the combination.

The *Data* field stores the instances of the combination according to the format expressed in the Format element. DCL4UTL proposes classical operators, such as arithmetical operators (+, -, *, /), logical operators (and, or, etc.), comparison operators ($<=$, $>=$, =, etc.) and predefined functions such as:

- *first* and *end*: give the first and the last element of a set.
- *index*: gives an element of a specific location in a set.
- *min* and *max*: give the min and the max for a set of numeric values.
- *sum*: evaluates the sum of all elements in a set of numeric values.
- *count*: gives the number of values in a set.
- *filter*: extracts a subset of values according to a specific criterium.

- *sort*: sorts the elements of a set ascending or descending.
- concat: concatenates two strings.

An interesting feature of DCL4UTL is the integration of external operators which can be operators or functions of other analysis tools, but also complex generic functions developed ad hoc. Sometimes, it is not possible to specify easily a transformation or a combination, because we need a particular algorithm (data-mining) or an operator which is available in a library (for instance, we needed a Java code analyzer for our experiments).



Indicator

Defining | Getting | Using

{ Component , Example , Calculation period }

Title :	Calculer la détection de l'existance de la classe Point
Component :	
Title :	PEDALO-CD-ClassDetail
Type component :	contentDatum
Component :	
Title :	PEDALO-RD-SelectionQuestion
Type component :	rawDatum
Description :	
Discussion :	
Example :	
Method :	{ Role involved }
Type :	Automatic
Tool :	{ Example }
Title :	Détection_existance_classe_Point
Location :	Tool.jar
Description :	
Text :	Calculer la détection de l'existance de la classe Point
Dcl4 utl formula :	

```
col (     $CDtmp=end(CD.* as sort(CD.time_recieve)) ;
          $t1=CDtmp.datum[time] ;
          $u=CDtmp.user ;   $s=CDtmp.session ;
          $e=CDtmp.datum[event] ;

            $cq=0;
            $numSQ=count(R.# as filter(R.user == u and R.session == s and R.#[T] <=t1) ) ;
            $ind=0;

            if(numSQ >1 )
             { if (e=="SQ"){ind=numSQ -1;} else{ind=numSQ;}
```

Figure 5. Editor screenshot for the getting section of an indicator

3.3. The UTL environment

The language UTL is specified in a conceptual model and various information models (one per type of data). In order to favor the sharing and re-use, we decided to provide a XML Schema of UTL and a generic data editor based on a XML Schema [15]. This editor (Figure 5) manages various roles (teacher, developer, and analyst) with more or less visibility on data. It is possible to put data descriptions inside a public set to favor re-use. We integrated a specific functionality for creating new data as fork of other ones. The editor integrates an "on the fly" validator in order to assist the analyst in describing data. The editor is developed in Ruby and available on the web. All data are stored in the same server with a XML database (eXist [16]).

UTL is able to evaluate indicators during the session and after the session, so you can use the editor to test your indicators directly on your tracks set (similar to a post session processing). It is directly connected to our DCL4UTL interpreter. We have developed an interpreter that allows executing DCL4UTL specifications. It is developed using JavaCC (Java Compiler Compiler), the parser generator used with Java applications. JavaCC reads the DCL4UTL grammar and converts it to a Java program that can recognize the matches between a formula and the grammar. The DCL4UTL interpreter is connected also to the eXist database. It reads the DCL4UTL source code and will execute the code if there is no syntax error in the code.

4. How to use UTL: three examples

In that section, we will present three ways of using UTL: (i) for re-engineering purpose by analyzing the learning situation after the session; (ii) for regulation purpose by providing indicators during the session; (iii) and for re-using purpose by assisting the teacher in his/her design of the learning situation.

4.1. Re-engineering

Before the learning session, people who involved in the role of teacher or pedagogical designer are interested in observing specific elements (activities, resources, interactions, etc.) in order to evaluate these elements and to improve the learning scenario for another session of the same course. They describe in their own language the indicators. The analyst specifies indicators with the UTL language. The session starts normally and the TEL system provides tracks. After the session, indicators are evaluated and proposed to the teacher and/or the pedagogical designer. According to these values, (i) they are satisfied and do nothing; (ii) they detect some problems and bring solutions inside their learning situation; (iii) they need more information and discuss with the analyst in order to build new indicators which are evaluated if possible, that is to say when all tracks needed have been provided by the TEL system. Finally, they change or not their learning situation.

4.2. Regulation

Before the learning session, people who involved in the role of teacher, pedagogical designer and/or tutor are interested in indicators to favor regulation during the learning session. They describe what is needed to observe and how, in their own language. The analyst specifies indicators with the UTL language. The session starts normally and the TEL system provides tracks. During the session, indicators are evaluated and values are given to the tutor. He/she adapts the learning scenario or interacts with students in order to correct the learning situation. Actually, it is possible to describe new indicators during the session, but we do not provide an "easy to use" editor for the tutor because, it is complicated to describe the data combination formulae.

4.3. Re-using

Before the learning session, people who involved in the role of teacher or pedagogical designer are interested in observing specific elements. They have access to all indicators proposed by their community and browse these indicators in order to find those interesting for their learning situation. They select and give them to the analyst who adapts the raw-data according to the new TEL system if necessary. The session starts normally and the TEL system provides tracks. According to their tracking purpose they will apply the use cases of section 4.1 and 4.2.

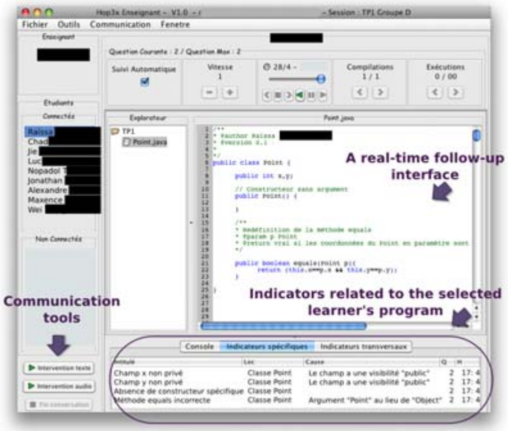
5. UTL in use

The UTL environment was used in 2010 for two learning sessions: a first session with six groups of fourteen first-year students in Multimedia, and a second session with two groups of eighteen third-year students in computer science. We proposed a practical activity of programming with a learning environment developed in our University.

This environment Hop3X is a track-based system which aids the supervision of real-time practical work in programming [17][18]. The tool allows teachers to monitor synchronously the activity of each learner in a group. Hop3X has a specific student interface (Hop3x-Student) which allows learners to edit, compile and run codes and programs. It also allows them to call teachers for help if needed. The teacher (or a tutor) can be in another location and has his/her own interface (Hop3x-Teacher cf. Figure 6) which offers a real-time visualization of learners' activities. Through a functionality of replay, it gives teachers the possibility to see again, during session, what learners did thanks to their collected activity tracks. The teacher is allowed to interact with student by means of a textual or an audio discussion. A visualization interface allows teachers to monitor indicators values related to each learner's activity. The Hop3X version with UTL indicators is now integrated by our University for the Oriented Object Programming course.

According to us, the main goal of these experimentations was to put to the test the language and the interpreter. Teachers have proposed around eighty indicators.

Because of our lack of "user friendly" editor, all indicators were formalized by a PhD student who works on tutorial activity. Some indicators were designed as global indicators such as the frequency of manual compilations, the percentage of private variables, etc. The most important set was designed according to specific questions such as for the first question: the existence of a specific class, the correctness of variables declarations, etc. We observed that our analysis environment succeeded in evaluating indicators, but that specific assistance has increased significantly the tutor's work. They intervened 4.5 times more than the previous year without indicators. The impact of indicators and tutor's reactions on students' learning is currently analyzed by researchers in tutoring systems.



```
/**
 * @author Raissa [redacted]
 * @version 2.1
 *
 */
public class Point {

    public int x,y;

    // Constructeur sans argument
    public Point() {

    }

    /**
     * Redéfinition de la méthode equals
     * @param p Point
     * @return vrai si les coordonnées du Point en paramètre sont
     */

    public boolean equals(Point p){
        return (this.x==p.x && this.y==p.y);
    }
}
```

Intitulé	Loc	Cause	Q	H
Champ x non privé	Classe Point	Le champ a une visibilité "public"	2	17: 4
Champ y non privé	Classe Point	Le champ a une visibilité "public"	2	17: 4
Absence de constructeur spécifique	Classe Point		2	17: 4
Méthode equals incorrecte	Classe Point	Argument "Point" au lieu de "Object"	2	17: 4

Figure 6. The monitoring interface of Hop3X with indicators visualization

6. Conclusion

This paper presents our proposal for a declarative and generic language for the specification of indicators in a Technology Enhanced Learning environment. It allows also the description of all data needed to evaluate an indicator.

We focus on a declarative language in order to capitalize them. We are interested in building a collection of re-usable indicators. They can be re-used directly with the same learning environment or adapted to a new one. For the adaptation, we have only to rewrite the link between UTL/P and UTL/T. All formulas are independent of the educational modeling language and the tracks format; they are based on the UTL information models.

We have developed an interpreter which can be connected with every systems and an editor to describe UTL data. We have successfully integrated the UTL environment in Hop3X, a learning platform of our university, and use it session. We will focus now on new enhancements such as the creation by the teacher of basic indicators (simple composition of indicators) during the learning session.

We are interested also on providing a recommender system during the instructional design process. By describing indicators as kind of patterns, we will be able to propose indicators and eventually improvements to teachers.

At present, we use the Format field, to describe how the result of the indicator has to be provided (a textual result). UTL will be augmented in order to categorize indicators and to associate one or more visualization methods (Graphical or not) with each category.

There is a technical limit on our interpreter: the use of the open source XML database. Queries may be too much slower than it is necessary. In our experiments, sometimes we had to evaluate more than fifty indicators (and their associated data) in less than one second. We will have to find some solutions to be as reactive as possible during the session.

7. Acknowledgements

We would like to thank C. Desprès and P. Jacoboni the authors of Hop3X, C. Choquet for the collaboration on the conceptual model of UTL, C. Lagrange for the development of the UTL editor and D. Pham Thi Ngoc who worked on the DCL4UTL and its interpreter during her PhD.

8. References

[1] Corbière, A. and Choquet, C., (2004), "Re-engineering method for multimedia system in education," in *IEEE Sixth International Symposium on Multimedia Software Engineering*, pp. 80–87.

[2] Corbière, A. and Choquet, C., (2004), "A model driven analysis approach for the re-engineering of elearning systems," in ICICTE'04, pp. 242–247.

[3] Lejeune, A. and Pernin, J.-P., (2004), "A taxonomy for scenario-based engineering," in *Cognition and Exploratory Learning in Digital Age*, pp. 249–256.

[4] Koper, R., Olivier, B., and Anderson, T., (2003), IMS Learning Design Information Model (version 1.0), IMS Global Learning Consortium, Inc. [Online]. Available: http://www.imsglobal.org/learningdesign/ldv1p0/imsld infov1p0.html

[5] Romero, C., Ventura, S., Pechenizkiy, M., and Baker, R. S., Eds., (2010), *Handbook of Educational Data Mining*, ser. Data Mining and Knowledge Discovery. CRC Press.

[6] Zendagui, B., (2010), "Support à la spécification des besoins d'observation dans un contexte de ré-ingénierie des scénarios pédagogiques : une approche dirigée par les modèles," Ph.D. dissertation, Université du Maine, France.

[7] Dimitracopoulou, A., Dillenbourg, P., Hoppe, U., and de Jong, T., (2004), ICALTS, "Interaction & collaboration analysis' supporting teachers & students' self-regulation," Tech. Rep. [Online]. Available: http://www.noe-kaleidoscope.org. Consulted: April 2009.

[8] Bratitsis, T., and Dimitracopoulou, A., (2005), "Data recording and usage interaction analysis in asynchronous discussions: The d.i.a.s. system," in *Workshop "Usage Analysis in Learning Systems"* AIED2005, pp. 17–24.

[9] Avouris, N., Komis, V., Fiotakis, G., Margaritis, M., and Voyiatzaki, E., (2005), "Logging of fingertip actions is not enough for analysis of learning activities," in *Workshop"Usage Analysis in Learning Systems"* AIED2005, pp. 1–8.

[10] Settouti, L. S., (2011), "Systèmes à base de traces modélisées - modèles et langages pour l'exploitation des traces d'interactions," Ph.D. dissertation, Université Claude Bernard Lyon 1.

[11] Choquet, C., Delozanne, E., Pozzi, F., Merceron, A., Oliveira, C., Luengo, V., Hotte, R., Beale, R., Verdejo, M.-F., and Stefanov, K., (2005) DPULS, "Design patterns for recording and analysing usage of learning systems," Kaleidoscope NoE, Tech. Rep. [Online]. Available: http://www.noe-kaleidoscope.org. Consulted: April 2009.

[12] Dimitracopoulou, A., Petrou, A., Martinez, A., Marcos, J.-A., Kollias, V., Jermann, P., Harrer, A., Dimitriadis, Y., and Bollen, L., (2005) IA, "Interaction analysis," Tech. Rep. [Online]. Available: http://www.noe-kaleidoscope.org. Consulted: April 2009.

[13] Schoonenboom, J., Keenoy, K., Montandon, L., Goita, Y., Faure, D., David, J.-P., Lejeune, A., Blake, C., Pluhar, Z., Kaszas, P., Jones, A., Turcsanyi-Szabo, M., and Levene, M., (2004), TRAILS, "Personalised and collaborative trails of digital and nondigital learning objects," Tech. Rep. [Online]. Available: http://www.noe-kaleidoscope.org. Consulted: April 2009.

[14] Pham Thi Ngoc, D., Iksal, S., Choquet, C., and Klinger, E., (2009), "Utl-cl: A declarative calculation language proposal for a learning tracks analysis process," in *Proceedings of The 9th IEEE International Conference on Advanced Learning Technologies* (ICALT' 2009), Riga, Latvia, pp. 681–685.

[15] Iksal, S., Pham Thi Ngoc, D., Lagrange, C., and Choquet, C., (2011). "Utl: A usage tracking language and the corresponding environment." [Online]. Available: http://eiah.univ-lemans.fr/UTL/UTL.xml. Consulted: March 2011.

[16] "exist: an open source native xml database." [Online]. Available: http://exist.sourceforge.net. Consulted: March 2011.

[17] Després, C., and Jacoboni, P. (2011). "Hop3x: a supervision tool for practical work in programming." [Online]. Available: http://eiah.univ-lemans.fr/HOP3X/HOP3X.xml. Consulted: March 2011.

[18] Lekira, A., Després, C., Jacoboni, P., Choquet, C., Iksal, S., Py, D., Pham Thi Ngoc, D., (2011). Using Indicators during Synchronous Tutoring of Practical Work. The *11th IEEE International Conference on Advanced Learning Technologies (ICALT'11)*, Athens – Georgia (USA), 6-8 July 2011