Usage analysis in an e-learning system: LD representation significance

Vincent Barré, Christophe Choquet, Sébastien Iksal, Alain Corbière

Usage Analysis in an e-Learning System: LD Representation Significance

Vincent Barré, Christophe Choquet, Alain Corbière and Sébastien Iksal
LIUM / IUT de Laval
52, rue des docteurs Calmette et Guérin
53020 Laval Cedex 9, FRANCE
FirstName.LastName@univ-lemans.fr

Abstract

In a pedagogical re-engineering process, it is important to be able to compare the effective system usage with prescribed scenario (which can be formalized with an Educational Modeling Language*). We think that computer tracks resulting from e-learning system usage make up pedagogical objects by themselves and thus they can be modeled with the help of an EML. In this paper, we will show that such an approach is interesting. We will more particularly focus on how to represent computer tracks with a specific EML (IMS Learning Design) and to highlight some cases in which this comparison of two LD scenarios (the prescribed one and the synthesized one) brings useful information to understand the effective progress of the learning session.*

1. Introduction

In the framework of MOCA project [1] we are interested in the pedagogical re-engineering of an e-learning system. This project falls into a normalization context (use of work done by IEEE/LTSC – http://ltsc.ieee.org/– *i.e.* LOM for description of pedagogical resources and LTSA for the architecture) and relates, amongst other things, to the definition of a pedagogical re-engineering methodology [4].

In order to do that, we have designed some experiments whose main goals were: (1) to determine the distance between learner activities and preconceived scenarios, (2) and to gather information on interactions between human actors (intervention strategies and content).

To investigate the first point, the prescribed scenario will logically be expressed in an educational modeling language (an EML) whereas the "scenario" of observed uses of the system will be derived from computer tracks obtained during the session.

We have chosen to use a specific EML to define prescribed scenarios: *Learning Design* (LD) from the IMS consortium [6] (which itself is derived from an EML introduced in early 2000 at the *Open University of Netherlands* by Koper [5]). This choice is in particular due to the fact that we want to associate ourselves to a normalization context. But also, and importantly, because LD has the most advanced specifications and several projects of LD editors or LD players are well advanced. Lastly, in the future LD will be closer to the SCORM specifications (http://www.adlnet.org/) which adds to its significance. A more detailed description of the LD language and a more explicit account of our motivations to choose LD can be found in [2]. We nevertheless think that it is interesting to briefly recall some characteristics of this EML.

From the LD specification designer's point of view, any pedagogical scenario can be modeled as a method specifying activities to certain actors and in a certain order. A scenario must at least contain a collection of *components* and a *method*. Components can be *roles* (learners, tutors… that can be split in many subgroups), activities or static scheduling of activities (activity-structures), whereas who (which role) does what (which activity) and at which moment is determined by the method (which, strictly speaking, can be considered as the scenario). A method is made up of one or many *plays* formed by a series of *acts*. Different plays will represent alternative scenarios while acts are subsets of the scenario that allow the activity synchronization for the different roles (all actors involved in an act must have finished it before starting the next one).

In the framework of a pedagogical re-engineering process, it is necessary to be able to "track" the activities of all actors of the system. These "observed uses" will be mainly used after a session to determine the distance between learner activities and preconceived scenarios, but can also be (at least partially) used during the session to help tutoring activities. In all cases, these computer tracks need to be represented in a format that will be usable by the designer or the tutor. This format will ideally be able to express most of the information contained in the computer tracks, whilst remaining "understandable" by designers or tutors.

Many track analysis methods linked to data-mining or more particularly web-mining (see for example [3]) can bring up information on the effective learner activity. These methods generally rely on mathematical techniques (variance analysis…) in order to identify patterns in data. These patterns are then used to support a decision making process (reorganization of a web site, definition new sales strategies…) with some explanations on user navigation history.

Data mining, and web mining generally deal with four major problems: segmentation (grouping individuals in homogeneous groups), association (identifying dependencies between observed characteristics), classification (explanation of a qualitative characteristic from other ones) and estimation (explanation of a quantitative characteristic from other ones).

But these methods are not specifically designed to bring together the observed activity and the prescribed scenarios (which are not generally specified in a formal way). Moreover, these tools do not integrate the specificity linked to the usage in a pedagogical environment.

Indeed, in the context of an e-learning environment, computer tracks obtained from learner activity are a little particular in that they contain a certain pedagogical semantic (for example, the observed route of a learner can be used to give feedback information on the effective learning). In this way, these tracks are true pedagogical objects containing information on learner activity during the whole session. It is thus natural to want to express them with an educational modeling language that allows the representation of the session dynamics.

In order to synthesize computer tracks, we have chosen IMS *Learning Design* but other EML languages (as defined by CEN/ISSS [8]) could also have been chosen (as long as the prescribed scenario and computer tracks synthesis are expressed with the same language). Such a representation with an EML has many advantages :

- computer tracks are expressed in a language that highlights its pedagogical semantics,
- using the same language to describe the prescribed scenario and the track synthesis allows us to compare them both and to bring a real meaning to the tracks,
- using the same environment to build prescribed scenarios and to explore computer tracks makes the designer's work easier.

We are presently working on such a tool that will construct LD scenarios with information contained in computer tracks arising from learner sessions.

2. Representing computer tracks with an EML

In order to build an LD scenario from computer tracks, we must be able to define activities done by actors and to indicate their scheduling.

So, we need to identify what has been done (delimiting activities) and who has done it (which learner made which activity) in the tracks.

Prescribed scenarios characterize the activity that will be found in computer tracks. The “components” section, from the prescribed scenario, contains all activities that are to be done by the system actors. These activities have to be singled out in the computer tracks by identifying corresponding patterns. Those patterns (for example the identifier of an activity) must be set up by scenario designer in activities description (e.g. using metadatas, even if LD does not actually propose a ‘natural’ way to do it). In addition, it is the LMS that add learner identification information to it’s tracks. With this information, we are then able to define activities done by actors and to indicate their scheduling.

An observed scenario (activity schedule) for a learner (or a tutor) can be represented by an *activity-structure* (in the *components* section, which is static) or by a play (in the *methods* section, which represents the dynamics of the scenario). These two choices are feasible in theory, but we must keep in mind that the reconstructed scenario must be expressed in a language that has the same “common meaning” as the source language. Plays allow us to indicate what should be done and who must do it. It is thus natural to prioritize their use (moreover it is mandatory to use a play to maintain the “common meaning” of the language, even if this play can use an activity-structure to describe the behavior of learner or tutors in a static way).

The next question that arises is whether to use one play per reconstructed scenario or alternatively one single play with as many role-parts as necessary (this binds an actor to an activity). LD specification points out that if many plays are present, they will be executed in parallel. Thus, this method is only interesting if we do not need to show the interactions between many learner activities (the interactions with tutors can be included in each play). Notice that if we have a single learner with no interactions with other actors (other learners or tutors) we may also want to use IMS *Simple Sequencing* [7] to express the computer tracks. On the other hand, if we want to be able to express potential interactions between learners or groups of learners in the reconstructed scenario, there must be only one play section in which many acts may possibly be present in order to synchronize actor activities (this will eventually lead to the creation of

new roles, different from those representing each individual learner).

Up to now, we have mentioned the idea of a "reconstructed" scenario, but there is not only one, but many, types of reconstructed scenarios:

- a scenario transcribing an individual track,
- a prescribed scenario "enhanced" (or decorated) with information gathered from computer tracks,
- an "observed scenario" that contains, for each activity, all observed continuations (eventually re-transcribing the interaction with other actors during the activity's progress).

It also will be interesting to add information in the prescribed scenario to point out elements to be tracked during a session (for example using LD properties and notifications). This information will be given to learners and especially to tutors (and in this case why not also use EML formalism to represent the useful computer tracks ?). Thus, the use of computer tracks will be done both during and after a learning session.

During a given session, we can obtain a "map" of activities done by learners (this can be also done by the learning management system). To do so, we should use the *on-completion* (action to be done when an activity is completed) property of an activity to set up a local property (*loc-property*) associated with this activity. This property can be used, for example, to determine number of learners having finished this activity. Then, during the session, it will be possible to consult this information with a *monitor* (LD mechanism allowing an actor to display its properties) before incorporating it in the scenario resulting from the computer tracks.

```
<loc-property identifier="P-AC1" >
  <datatype="integer" />
  <initial-value="0" />
</loc-property>
```

definition of a new property

```
<learning-activity identifier="LA-AC1" >
  <on-completion>
    <change-property-value>
      <property-ref="P-AC1">
      <property-value>
        <sum>
          <property-ref="P-AC1" />
          <property-ref="P-increment" />
        </sum>
      </property-value>
    </change-property-value>
  </on-completion>
</learning-activity>
```

one more learner has completed this activity

After a session we will construct scenarios "enhanced" by the gathered information. For example, the "observed scenario" will be assembled in the following way:

In the *components* section we will find a list of all prescribed activities, each one associated with a *loc-property* representing the number of learners having completed this activity.

We will use an *activity-structure* in order to show all observed continuations after an activity (associated with observed percentages during a session).

For example, if we extract the following information from computer tracks:

	AC1	AC2	AC3	AC4	AC5	AC6
AC1		25%	75%			
AC2						
AC3				33%	66%	
AC4						
AC5						25%
AC6				5%		

we can incorporate this information into LD scenario in the following way:

```
<loc-property identifier="P-suites-AC1-AC2" >
  <datatype="real" />
  <initial-value="0.25" />
</loc-property>

<activity-structure identifier="AS-suites-AC1"
       number-to-select="1" structure-type="selection">
  <learning-activity-ref ref="LA-AC2" />
  <learning-activity-ref ref="LA-AC3" />
</activity-structure>

<activity-structure identifier="AS-AC1"
       number-to-select="2" structure-type="sequence">
  <learning-activity-ref ref="LA-AC1" />
  <activity-structure-ref ref="AS-suites-AC1" />
</activity-structure>
```

The preceding scenario does not take into account potential interactions between learners or between learners and tutors. Nevertheless, it can be interesting to take into account the modifications arising from those interactions. One could, for example, want to express the following situation:

- LA-AC1 activity done alone $\Rightarrow$ 25% of learners intend to continue with activity LA-AC2,
- LA-AC1 activity done together with support activity SA-AC1 $\Rightarrow$ 66% of learners intend to continue with activity LA-AC2.

Such behavior will be considered in the *method* section (that is, in the dynamics of the scenario) using conditions that allow the modification of properties associated with observed continuations of an activity:

```
<method>
  <play>
    <act>
      <role-part>
        <role-ref ref="R-apprenant" />
        <learning-activity-ref ref="LA-AC1" />
      </role-part>
      <role-part>
        <role-ref ref="R-tuteur" />
        <support-activity-ref ref="SA-AC1" />
      </role-part>
      <on-completion>
        <change-property-value property-ref="P-suites-AC1-AC2"
             property-value="0.66" />
      </on-completion>
    </act>
  </play>
</method>
```

It will also be instructive to consider the order in which activities are done so as to highlight the influence of this order on the observed continuations of an activity.

From this point of view, a more general problem is how to represent the order in which activities are done. This representation is natural when we are constructing the scenario followed by a particular learner, but it is

also relevant to supply synthesized information such as: for a given activity, at what precise moment is it done by learners?

In the particular framework of an evaluation activity (such as a multiple-choice questionnaire), it is particularly valuable to keep scores obtained by learners (for example to see if *when* an evaluation is done has an influence on its results). Thus, it will be helpful to represent information such as:

x% of learners are doing evaluation as the 1st activity with an associated success rate of x'
y% of learners are doing evaluation as the 2nd activity with an associated success rate of y'
z% of learners are doing evaluation as the 3rd activity with an associated success rate of z'
…

A worthwhile solution would be to use an array to associate this information with an activity. Unfortunately, the LD specification does not allow this (properties must be of a simple type, not arrays). Nevertheless, one can create as many properties as necessary, but it is not the best solution.

Another typical configuration that can be extracted from computer tracks is the "loops" in the scenario of an individual session. A loop in the observed scenario can be typical of "normal" learning (if the learner needs to go backwards in order to understand something better), but a loop can be also an indication of a problem in the activity scheduling or of the malfunctioning of the activity. It is thus fundamental to be able to return this information to designers or tutors. Such a loop can be represented by an *activity-structure* having the same activity at the beginning and at the end of the sequence.

Moreover it would be informative to associate different information to activities at each repetition of the loop (for example the success rate in an activity or the score obtained in a test). Ideally, this information must be stored at each repetition of the loop, as specified by the designer.

To do this, it would again be useful to have array properties.

3. Prescribed scenarios, reconstructed scenarios and comparison perspectives

Another advantage linked to the synthesis of a "scenario" with the help of the same EML as the one used for expressing the prescribed scenario, is that it is possible (and useful) to compare them. This comparison enhances information gathered from computer tracks and will be valuable to the designer for improving the scenario.

Let us take an example from our experiments. Our scenario prescribed six activities that learners were free to tackle in any order [1]. Nevertheless, the designer conceived one "predicted" scenario, and this scenario included a multiple-choice questionnaire whose main goal was to evaluate knowledge acquisition. After our first experiment, some computer tracks revealed that only a few learners successfully completed the questionnaire, which was surprising. The reason for this "failure" was that a relatively important number of learners used this questionnaire in order to evaluate *a priori* their knowledge but not *a posteriori* as it was intended by the designer. One way to focus on this fact in the reconstructed scenario would have been to consider information such as: « *x%* of learners have done the questionnaire as intended in the prescribed scenario, with an associated success rate of *x* but *y%* (significant) have done it differently with a success rate of *y* ». Moreover, it would be interesting to be able to itemize those results.

In order to do this, we can use the preceding assemblies. First, we need to define how many learners completed the questionnaire as the first activity (with the associated result), as well as how many learners completed the questionnaire as the second activity (with the associated result), and so on. Next, we need to determine a sub-sequence of the prescribed scenario and observe its implementation (for example, our questionnaire evaluating a learning activity). Lastly, we need to identify this sub-sequence in the graph synthesized from the computer tracks, using information on the order in which activities are done in conjunction with graph of the observed continuations.

Another use of computer tracks in understanding the "failure" of an activity (whether it evaluates or not) is to highlight the learners effective route through this activity and to compare it with the anticipated one. We can use both the observed continuations graph and the loops observed in synthesized scenario to do this.

In our experiments, learners had to devise a Java application, compile it and test it. We noticed that some learners had an abnormally low success rate for the "application testing" activity. Use of synthesized information from computer tracks has shown that those learners were generally following a scenario that looped on the activities edit, compile, test, edit… Such a loop can be the result of programming mistakes, but it was not relevant here (a tutor was available). In fact, in order to "understand" this problem, we had to look deeper into the computer tracks to discover that those particular learners had not saved their source code before compiling it (and so they were constantly compiling the same faulty code).

In some cases, the use of the synthesized scenario alone is thus not sufficient to "understand" the problem. That is particularly the case when the problem involves activities which are not specified in the prescribed scenario (for example because they are implied).

4. Conclusion

In a re-engineering cycle, highlighting relevant information obtained from computer tracks is of prime importance. From this viewpoint, in this paper we have proposed a way to synthesize pedagogical information obtained during a learning session by using an educational modeling language identical to the one that describes the prescribed scenario. We think that using the same language is strategic at several different levels: it is easier for the designer to understand the observed uses of the e-learning system, we have a natural way to compare prescribed scenarios to observed uses, and using an EML highlights the pedagogical information embedded in computer tracks.

We are presently developing an analysis tool that will allow us to synthesize a set of LD scenarios using computer tracks and the prescribed scenario.

From this viewpoint, LD specification allows us to reconstruct scenarios using observed uses (as they are shown through computer tracks). Nevertheless, the design of our tool puts the focus on some points that could be enhanced in LD specification, which in turn would improve the reconstruction of observed scenario and their usability. For example, how to deal with problems arising from the fact that properties in LD must be simple data types (even though arrays would be useful). Another example is that nothing is provided to allow the designer to indicate interesting information to track. Lastly, there is nothing in the LD specification that offers a natural solution to the problem of finding activities from the prescribed scenario in computer tracks.

Nevertheless, we think that using an EML to express information found in computer tracks is valuable for designers and that IMS LD already has a number of qualities to do this.

5. References

[1] V. BARRE, Ch. CHOQUET, A. CORBIERE, Ph. COTTIER, X. DUBOURG & P. GOUNON, “MOCA, une approche expérimentale de l’ingénierie des EIAH”, Environnements Informatiques pour l’Apprentissage Humain, Strasbourg (France), pp. 55-66, 2003.

[2] V. BARRE, “EMLs : case study in distance learning education”, CALIE’04, Grenoble (France), pp. 155-160, 2004.

[3] M. BAZSALISCZA & P. NAIM, *Data Mining pour le Web*, Editions Eyrolles, Paris, 05/2001

[4] A. CORBIERE & Ch. CHOQUET, “Designer integration in training cycles: IEEE LTSA model adaptation”, CALIE’04, Grenoble (France), pp. 51-62, 2004.

[5] R. KOPER, “Modeling units of study from a pedagogical perspective : the pedagogical meta-model behind EML”, 06/2001.

[6] R. KOPER, B. OLIVIER & T. ANDERSON eds., *IMS Learning Design Information Model*, IMS Global Learning Consortium, Inc., version 1.0, 20/01/2003.

[7] M. NORTON & A. PANAR eds., *IMS Simple Sequencing Information and Behavior Model*, IMS Global Learning Consortium, Inc., version 1.0, 03/03/2003.

[8] A. RAWLINGS, P. ROSMALEN, R. KOPER, M. RODRIGUEZ-ARTACHO & P. LEFRERE, *Survey of Educational Modelling Languages (EMLs)*, CEN/ISSS WS/LT Learning Technologies Workshop, version 1, 19/09/2002.