Observation Scenario Development Using Recommendations

Vincent Barré, Christophe Choquet, Sébastien Iksal

Observation Scenario Development Using Recommendations

Vincent BARRE, Christophe CHOQUET and Sébastien IKSAL
LIUM / IUT de Laval
52, Rue des docteurs Calmette et Guérin
53020 LAVAL Cedex 09 / FRANCE
FirstName.LastName@univ-lemans.fr

Abstract

Within in a re-engineering context, we consider that the designer of a pedagogical situation is the actor who is the most able to describe his observation intentions. This description must be done at the same time as the pedagogical scenario development.

In this paper, we present a process that does not rely on the educational modeling language used by designers and which aims to support them in their observation scenario modeling task.

This process relies both on a language, UTL (Usage Tracking Language), which allows the description of tracks that they wish to obtain and their linking with tracking objectives associated with used language concepts, and also on the introduction of filters. Once applied to the pedagogical scenario under construction and to the used language model, these filters open up potentially interesting tracking possibilities, that we will call "recommendations".

1. Introduction

The very nature of learning and distance teaching applications brings about de-synchronization of teachers' two major roles: the instructional designer who sets up learning goals and learning situations and the tutor who regulates the learning process.

Compared with student centered works, few works deal with supporting the designer, especially in a re-engineering process. Designers' support is usually *ad hoc* and not reusable, for many potential reasons: (i) the instructional designer's point of view (e.g. the design model) is not sufficiently explicit, (ii) students' tracks come down to log files conditioned by the software abilities, (iii) potential problems in the interaction sequence are not previously identified, (iv) solutions adopted are not well defined and recorded, (v) teaching actors (e.g. instructional designers and tutors or teachers) do not systematically communicate, (vi) context and usage are not well defined and analyzed.

But each computer-based learning project needs to track students and to analyze their activity in order to adapt the didactic method dynamically during a session and/or to modify contents, resources and scenarii after the session, to prepare the next one. Considerable research has been done already in this field [7][8], especially to help teachers to adapt the pedagogical scenario or to help students to self-regulate the learning process. There is a large amount of scattered research expertise about techniques to collect and analyze data in technology supported learning activities (see [4] for a state of art). Research actions recently lead by Kaléidoscope[1], an European network of excellence, as well as the worldwide emergence of an Educational Data Mining community[2], show the scientific community's interest in learning activity tracking and analysis.

The REDiM project (French acronym for Model Driven Reengineering of a TEL system) [2] tackles this topic by refocusing TEL system tracking and usage analysis modeling on pedagogical scenario designers.

We consider that learning activity tracking is only interesting within a pedagogical context, either in order to control the activity, or to improve the original scenario in a re-engineering process. That is why we want to bring pedagogical designers to clarify their observation intentions from a pedagogical viewpoint. We therefore encourage the designer to model the learning session observation scenario whilst modeling the pedagogical scenario. We support this activity by offering a language, UTL (Usage Tracking Language), which allows designers to describe the tracks (called indicators) that they wish to obtain and to link them to tracking objectives associated to

1 http://www.noe-kaleidoscope.org/

2 http://www.educationaldatamining.org/

concepts from the pedagogical scenario representation model (see section 4 for more details).

So as to leave it to designer to choose their educational modeling language (EML), we have designed UTL to be a meta-language that can be instantiated on the EML currently used. Moreover, in order to support the designer in their track modeling task, we have set up filters that, once applied to the pedagogical scenario under construction and to the used EML model, provide potentially interesting tracking possibilities that we call "recommendations". These tracking recommendations are proposed to the designer who, if he thinks that they are relevant, can associate them to a tracking objective, using our UTL language, and therefore make them indicators. In this case, our filters also contain some hints on how to represent indicators in the pedagogical scenario.

We have already shown in [1][3] the interest and feasibility of this approach when IMS Learning Design is used to model pedagogical scenarios. We prove in this paper that this approach is also interesting and feasible in another context, that is, in a context where the pedagogical scenario representation model is specific and non-standard.

After a short presentation of the LEA project and of the associated pedagogical scenario representation model, we will come back to clarifying the recommendations and filters used to suggest potentially interesting tracks. Then, we will see how to formalize those tracks using UTL and we will present a working example of this process.

2. LEA project presentation

The LEA (French acronym for Apprenticeship Electronic Booklet) is a Learning Management System designed specifically for supporting apprenticeship. It is the result of a research and development project conceived as a participatory design process. LEA is a web-based system focused on the assessment and the regulation of apprentices' learning, which offers different functions to many actors (the training center, the company, the apprentice, and even his parents). The LEA conception phase has brought trainers (at the same time designers and teachers) to clarify use scenario they have anticipated for LEA; the Web application functions are in fact the elements that have emerged from those scenarios.

In the LEA project, the EML used to formalize the scenario is not *a priori* fixed, but rather negotiated between designers in the project. More precisely, designers develop jointly their EML and the instances they are modeling, so as to better fit their pedagogical purpose [5].

In this paper, we will focus more particularly on a model established by one participatory design group and depicted in figure 1.

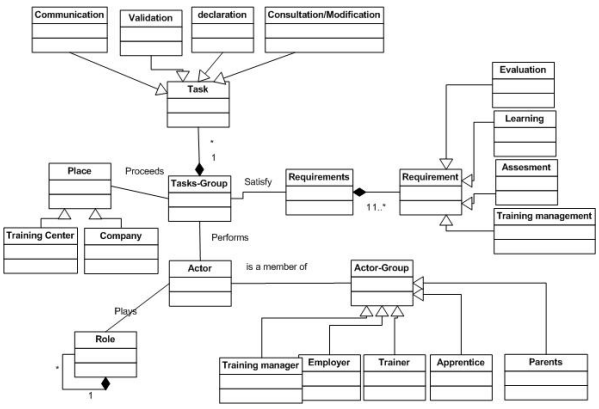

Figure 1. A model used in the LEA project

We will now apply our recommendation mechanism to this model (and, more precisely, on the XML Schema transcribing it).

3. Tracking recommendations

Our goal is to suggest potentially interesting tracks to designers during the scenario modeling phase. The main idea is to let them elaborate their scenarios / models and to propose a mechanism allowing them to consult our recommendations whenever they want. They can thus consult these recommendations, filtering them with rules introduced in [1], to define whether: (i) the recommendation is useless, (ii) it corresponds to a potentially useful data or (iii) it corresponds to pedagogically interesting and significant data (e.g. an indicator).

In [1], we have introduced six "rules" of use to identify and filter "recommendation candidates". Our operation process is improved by better formalizing the candidates and by identifying the way they instanciate themselves on designers used model. Afterwards, *generic recommendations* will be useful to identify recommendation candidates that can, in turn, be filtered out by our *filters*.

Recommendations, as well as filters, are linked to the designers defined EML XML Schema. They activate themselves on the *condition* satisfaction they contain, that is, depending on XML type of language element. The *representation* contains indications on changes that need to be brought to the XML Schema to incorporate recommendations.

Generic recommendation #1

Condition : XML type = *complexType*
Recommendation : calculation of the number of instances found in tracks
Representation : meta-model alteration, adding an *observedOccurs* attribute

This recommendation can be applied to *LEA-Model*, *Actor*, *Requirements*, *Tasks-Group* and *Task* elements, although it is only really interesting for the *Actor* element.

Generic recommendation #2

Condition : XML type = *simpleType*
Recommendation : transcribe values of this element (taking into account its different values, if applicable)
Representation : extend considered element *simpleType* to add a new attribute' list, which allows to transcribe value alterations in a chronological way.

This recommendation can be applied to the *Task-Type* element, which is a *string*. Moreover, many other elements are in the same case: *Requirement-Type*, *Actor-Group-Type*, *Place-Type*, *Title* and *Name*.

Generic recommendation #3

Condition : references or contains another element for which we have determined tracks
Recommendation : import those tracks into element context
Representation : use representation method described in [1].

This recommendation can be applied to the *Tasks-Group* element that contains a *Task* collection consisting of a *Title* and a *Task-Type* (simple elements for which recommendation #2 proposes value location). Then, recommendation #3 allows the linking of these two elements to a *Task* that can, in turn, be linked up with other *Task*s in a *Tasks-Group*.

These recommendations can then be filtered out before being presented to designers. We highlight here a filter taken out from [1] and adapted to the design model used in the LEA project.

Filter F-1

Condition : XML type = *complexType* & element sequence with an unbounded *maxOccurs* property
Action : locate unused elements
Representation : use the same representation as for generic recommendation #1

This filter can be applied to *Requirements* element that contains a collection of *Requirement,* or to *Tasks-Group* that contains a collection of *Task*.

Proposed filters or generic recommendations have a *Representation* item, used to specify how to transcribe tracks that have been retrieved. One can notice that this method is particularly suitable whenever designers build their own languages since our process implies they modify their modeling language to incorporate information coming from the learning session (which is rather a problem in a normalized context).

4. Tracking formalization

In the framework of our activities, centered on model driven reengineering of a TEL system [2], we are interested in tracking and more precisely, in the definition of a language dedicated to the description of the tracks and their semantics. This language, UTL (*Usage Tracking Language* [3]), allows the definition of the observation needs and the means required for data acquisition. It allows the structuring of tracks, from raw data – those acquired and provided by the TEL system during the session – to indicators [6] which mean something significant for its users, *e.g.* designers. Its conceptual model is depicted on figure 2.

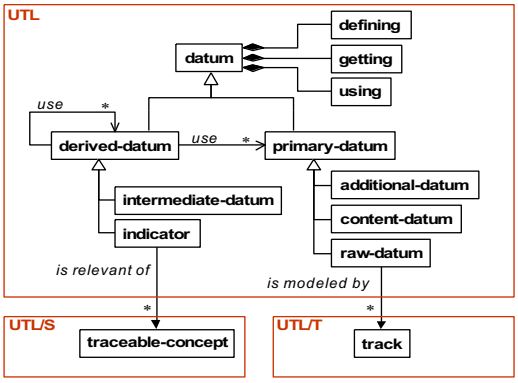

Figure 2. UTL conceptual model

UTL uses three facets to describe data: the *defining* one which models data meaning, that is the observation needs, the *getting* one which models the tracks means, and the *using* one which represents data value and therefore models the tracks uses. These three facets are both useful to prescribe tracks before a session, and to calculate data during or after a learning session.

There are two kinds of data: *primary* or *derived*. Primary data can be either *raw data*, coming directly

from computer tracks; *content data*, that is TEL system user products; or *additional data* which comes from session context, for example the learning scenario. This data does not arise from a calculation or a combination of other data. Derived data, on the other hand, is calculated using a method described in their getting facet. This data is necessarily obtained using other data, either derived or primary. The difference between *intermediate datum* and *indicator* is in its associated pedagogical semantic. An indicator is always relevant in a pedagogical context, it is systematically associated with a tracking objective and therefore, with an exploitation purpose for designers. Thus, it is associated with at least one concept from pedagogical scenario representation model. Concepts that can be associated to indicators are called *traceable concepts* and must be, above all, qualified by designers as traceable during the scenario elaboration.

Therefore, the UTL language is linked to our recommendation mechanism in that way. First, generic recommendations suggest potentially interesting intermediate data. Then, designers analyze these recommendations and can decide that this intermediate datum is a good indicator candidate. In that case, they must complete data description to make it persistent (that is, described with UTL and potentially reusable). In order to do that, designers will mainly have to define pedagogical semantic associated to this intermediate datum.

5. Use case

We will consider here that designers are modeling the following situation: an apprentice which is in a company period must declare activities done during this period. This evaluation should be filled out throughout the period and validated at the end.

This situation can be modeled with scenario depicted in figure 3, expressed in LEA model (cf. figure 1). We will use this situation to detail our recommendation process.

```
<LEA-Model N-Order="1">
    <Name>Apprentice view – Pharmacy</Name>
    <Place>Company</Place>
    <Actor>
        <Role>Apprentice</Role>
        <Actor-Group>Apprentice</Actor-Group>
    </Actor>
    <Requirements>
        <Requirement>Training management</Requirement>
        <Requirement>Learning</Requirement>
    </Requirements>
    <Tasks-Group>
        <Task N-Order="1.1">
            <Title>Declaration period choice</Title>
            <Type>Declaration</Type>
        </Task>
        <Task N-Order="1.2">
            <Title>Tasks carried out during the period</Title>
            <Type>Declaration</Type>
        </Task>
        <Task N-Order="1.3">
            <Title>Tasks themes declaration</Title>
            <Type>Declaration</Type>
        </Task>
        <Task N-Order="1.4">
            <Title>Tasks frequency declaration</Title>
            <Type>Declaration</Type>
        </Task>
        <Task N-Order="1.5">
            <Title>Declaration validation</Title>
            <Type>Validation</Type>
        </Task>
    </Tasks-Group>
</LEA-Model>
```

Figure 3. LEA pedagogical scenario

5.1. Recommendations development

This scenario is modeled with an *LEA-Model* element, for which generic recommendation #3 can be applied and refers to potential recommendations that can arise from its sub-elements: *Name*, *Place*, *Actor*, *Requirements* and *Tasks-Group*. We therefore need to examine those five sub-elements and to merge all recommendations arising from them in order to build a recommendation associated to the *LEA-Model* element.

Name and *Place* are of simple type *string*, generic recommendation #2 specifies recording their values.

Actor element is made of a *Role* / *Actor-Group* collection, generic recommendation #1 therefore leads us to consider the number of users in the session. Moreover, recommendation #3 can be applied to its two sub-elements: *Role* (complex type) and *Actor-Group* (of simple type, recommendation #2 thus indicates to record the value). This recommendation then suggests merging data arising from those two sub-elements and associating them to the *Actor* element. Then, we must clarify recommendations linked to *Role* element. Such an element is made of a *Title* (of simple type, recommendation #2 thus indicates to record the value), of a *Role-Description* (of simple type, recommendation #2 thus indicates to record the value), and of a *Role* element (recursive definition, recommendation #3 thus indicates to recursively merge data arising from *Title* and *Role-Description*).

One can break down in a same way *Requirements* and *Tasks-Group* elements using our recommendations.

The aggregate of all these recommendations thus constitutes one data, merging information on actors of the session (from *Actor* element), tasks executed or not (from *Tasks-Group* element), temporal sequencing of the session (from recommendation #2 applied on many elements)…

Finally, designers need to decide whether they want to include this recommendation to their scenario or not. If they decide to consider it, they can choose to make it an indicator, which needs to be formalized.

5.2. Designers validation

Once recommendations have been recursively applied on the scenario elements, the resulting data is proposed to designer. In our example, and following these recommendations, the designer wishes to observe a special user: the apprentice. The tracking objective consists of understanding apprentice behavior concerning declarations he makes during the time spent in his company. More precisely, the designer is interested in determining whether apprentices regularly fill out their declarations, throughout their placement in the company.

The designer therefore indicates that the recommendation arising from our process is an intermediate datum (from an UTL viewpoint) that corresponds to "declaration input regularity during company periods". It is actually about a task list with specific completion dates and beginning and end dates of the company phases.

The designer also thinks that it is possible to build an indicator from this data. From this viewpoint, he decides to qualify task sequence as: *missing* when declarations are not filled out; *steady* when the sequence follows the designer's expectations; *focused on the end*, when apprentice has filled out his declaration, but in one go and at the end of company period; or *made at the training center*, when declaration is filled out after the company phase. Moreover, the designer identifies two traceable concepts: *Actor* and *Task*.

This new data then forms an indicator; both because it is semantically relevant to assess apprentice activity, and to consider scenario re-engineering. Indeed, designer can also use this indicator to adapt scenario to encourage apprentices to complete these declarations, for instance including checkpoints.

6. Conclusion

In this paper, we have presented a process supporting designers in their observation modeling task. This process relies on generic tools, working both on the scenario being edited and on language used to elaborate this scenario, allowing the recommendation of potentially relevant tracks.

We had already shown the interest and feasibility of this approach when IMS Learning Design is used to model pedagogical scenarios. We have proven, in this paper, that this approach is also interesting and feasible in another context, that is, in a context where the pedagogical scenario representation model is specific and non-standard.

Another key point of our process is that it allows designers to define their tracking objectives using recommendations arising from it. Indeed, for each recommendation, they can specify the UTL characterization of this data and associate it to a tracking objective if they think it is relevant, which in turn allows it to become an indicator.

Indicators thus underlined are linked to the designer's used model. They can be capitalized and proposed to all designers using the same model to enrich their tracking possibilities.

Moreover, when a designer describes an intermediate datum as an indicator, he assigns it a pedagogical semantic. So, using links between this intermediate datum and model elements makes embedded semantic of the designer's model more explicit. We think that this clarification will lead, in the future, to an enrichment of our recommendations by taking into consideration the semantics of model elements on which generic recommendations can be applied.

7. References

[1] Barré V., Choquet C. (2005), *Language Independent Rules for Suggesting and Formalizing Observed Uses in a Pedagogical Reengineering Context*, In: Proceedings of ICALT'05, July 5-8, 2005, Kaohsiung (Taiwan), p. 550-554.

[2] Choquet C., Corbière A. (2006), *Reengineering Framework for Systems in Education*, In: Journal of Educational Technology and Society, Vol. 9 (4), p. 228-241.

[3] Choquet C., Iksal S. (2007), *Modeling Tracks for the Model Driven Reengineering of a TEL System*. In: Journal of Interactive Learning Research, Vol. 18 (2), Edited by AACE.

[4] *Design Patterns for Recording and Analysing Usage of Learning Systems* – Deliverable 3, D32.3.1 : State of art of tracking and analysing usage, Kaléidoscope NoE (2005).

[5] El-Kechaï H., Choquet C. (2006), *Understanding the Collective Design Process by Analyzing Intermediary Objects*, In: Proceedings of ICALT'06, July 5-7, 2006, Kerkrade (The Netherlands), p. 1047-1051.

[6] *Interaction & Collaboration Analysis' supporting Teachers & Students' self-regulation,* Kaléidoscope NoE (2004).

[7] Merceron A., Yacef K. (2004), *Mining student data captures from a Web-based tutoring tool : Initial exploration ans results*, In: Journal of Interactive Learning Research, Vol. 15 (4), p. 319-346, Edited by AACE.

[8] Mostow J. (2004), *Some useful design tactics for mining ITS data*, Proceedings of the ITS'04 Workshop on

Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, p. 20-28.