Visualization of Learning Scenarios with UML4LD

Pierre Laforcade
University of Le-Mans
FRANCE

Abstract
Present Educational Modelling Languages are used to formally specify abstract learning scenarios in a machine-interpretable format. Current tooling does not provide teachers/designers with some graphical facilities to help them in reusing existent scenarios. They need human-readable representations. This paper discusses the UML4LD experimental research work in relation to the graphical representation of abstract learning scenarios. We discuss the benefits teachers/designers can expect to reach as well as some scientific and/or technical obstacles researchers have to overcome to realize such models transformation. Our experiment concretely concerns the automatic generation of UML activity diagrams from IMS-LD learning scenarios.

Keywords
learning scenarios, instructional design, model driven engineering, UML , model transformation.

Introduction

Present *Educational Modelling Languages* (EML) (Kinshuk et al., 2006) are used to formally describe abstract learning scenarios, ensuring by this way reuse, exchange and interoperability (Koper, 2006) over several *Learning Management Systems* (LMS). The *Learning Design* specification (IMS, 2003a), from the IMS consortium (IMS-LD), is the current standard for these EML. The instructional design process proposed in the LD *Best Practices* (IMS, 2003b) of the IMS-LD specification follows three steps:

1. A concrete educational problem is analysed, usually between various stakeholders. The analysis results in a didactical scenario that is captured in a *narrative*, often on the basis of a *checklist* (analysis phase).

2. The narrative is cast in the form of a *UML* (Unified Modelling Language) *activity diagram* in order to add more rigor to the analysis (first design step).

3. The UML activity diagram then forms the basis for a XML document instance that conforms to the IMS-LD specification (second design step).

The resulting LD-scenarios are formatted by means of an XML binding in order to be interpreted by machines. Nevertheless, it is difficult to reuse, share and understand such formatted-scenarios by humans, without being aware of the associated narration and/or the UML activity diagram. Indeed, the UML activity diagram is a well-known means of visually representing and describing who (roles) does what (activities) and when (sequencing of activities): this work-flow representation is considered in the instructional design context as a learning-flow representation (Martínez-Ortiz et al., 2005).

Current Learning Design tooling (De Vries et al., 2006) still focuses on IMS-LD editors that directly produce XML documents conforming to the specification, for example see the formula-based Reload Editor (Reload, 2007). Some initiatives are appearing related to graphical design editors providing designers with export facilities towards the standard as for example the 'Mot+LD' editor (Paquette et al., 2006). Some current learning design research initiatives are also focusing on domain-specific instructional languages (Botturi et al., 2007), but there is still a real need for user-friendly tools with graphical user interface and for tools focusing on educators as end-users and not only trained designers. By analogy with software engineering principles, the reverse-engineering of such formal scenarios is never tackled in spite of the potential value-add it can provide by enhancing understanding and reuse of scenarios.

The research work presented in this paper deals with the graphical representation of abstract learning scenarios previously specified by means of an EML. We aim to provide teachers/designers with dedicated tools that can visually represent abstract scenarios. Throughout this paper, we use the descriptor “teacher/designer” to address both roles of teachers and designers for the same actor. Indeed, in our research we do not consider instructional design in an “industrial” approach (with various stakeholders such as pedagogues, instructional designers, learning contents providers, etc.) but in a more “traditional” way: teachers are those who “design” the unit of learning in terms of learning activities and learning content towards pedagogical and didactic objectives. Dedicated learning design tools will help teachers/designers to have a better understanding of the scenarios, insuring in this way the reuse and the exchange of abstract scenarios by neophyte teachers. This context is particularly interesting when these teachers want to integrate Technology Enhanced Learning in their practice and when they are lacking some experience in using formal approaches like IMS-LD.

In opposition to current approaches based on various research domains - design patterns, ontologies, semantic web, etc. (see Koper, 2006) for a global overview of current research in Learning Design), our work takes place within a Model-Driven-Engineering (MDE) context (Kent, 2002; Favre, 2004). We explain in Laforcade et al. (2006), the interests and potential benefits for the application of the principles and techniques of such a software engineering approach to the learning design context. In short, the learning scenario is the scientific and central model of this approach that consists of providing concrete services and tools in order to describe scenarios at a teacher level (with their vocabulary and semantics), to specify them at an abstract level (the current one tackled by the EML research works), and especially to transform the scenarios between each one of these various representations.

This paper concretely aims at: i) presenting and discussing the models transformation between abstract scenarios (specified with an EML) and domain-specific scenarios (human-readable), and ii) presenting the technical process we have experimented with, to demonstrate the benefit of a graphical representation, automatically generated, from an abstract learning scenario. This experiment is concretely centred on the transformation of XML-based IMS-LD-models towards UML activity diagrams. A specific language and tooling (*UML4LD*, i.e. UML for the IMS-LD specification) has been created.

The next section of the paper presents the research context and details our terminology. We then describe the technical process undertaken to illustrate the experiment. Finally, we discuss the highlighted benefits as well as the obstacles of such transformations.

Model Driven Engineering research context

This research takes place within a more general REDIM project (the acronym stands for *Model-Directed Re-engineering of Technology Enhanced Learning*). This project focuses on the formalization of re-engineering processes for learning scenarios (Laforcade et al., 2006); it highlights and deals with these topical issues:

- improving the reuse of learning scenarios;

- capitalizing the knowledge and sharing the experiences between teachers/designers;
- enhancing and supporting learning scenario design by providing teachers/designers with techniques and tools for the retro-conception and re-engineering (Chikofsky et al., 1990) of their systems.

Abstract Scenarios and Domain-Specific Scenarios

This paper is only focusing on the design phase and not the deployment or runtime phase of learning scenarios. We define a domain-specific scenario as a learning scenario described in a human-readable way addressing teachers/designers. Visual formalism can be textual as well as graphical. The very first objective of such a scenario is to explicitly describe the mental representation of the learning situation during design, at a *knowledge level* (Newell, 1982). In addition, domain-specific scenarios facilitate understanding between the actors of the pluri-disciplinary design team, and, in this way, they serve as a support to thinking. A domain-specific modelling language can be more or less formal, and more or less operational (i.e. having an optional XML binding). The vocabulary (concepts and relations) is that of the teachers/designers community. Its pedagogical expressiveness is generally specific to a pedagogical approach or other learning specificities. CPM (Laforcade, 2005), MISA/MOT (Paquette, 2004) are examples of such languages.

On the other hand, an abstract scenario is firstly specified and formatted into a machine-readable/interpretable way. Its first objectives concern its reuse, exchange and interoperability among a wide instructional design community of practice. In addition, the associated modelling languages, generally those mentioned by the 'EML' acronym, cover several instructional theories by mean of a very abstract and conceptual vocabulary. Their abstractness is related to both the domain-level and the LMS-independence. Such languages are formally operational (because of their first objectives) but graphical representations of abstract scenarios can also be interesting to better understand and handle them (works on MOT+LD go into this (Paquette et al., 2006): the MOT language has been extended by introducing the IMS-LD concepts with the notation of MOT). IMS-LD is an example of such EML.

Towards Learning Scenarios Transformations

The previous separation between abstract and domain-specific scenarios becomes immaterial if every learning design community develops its own domain-specific modelling language, even if all teachers/designers use the same abstract standard. In our opinion, tools are needed to help the definition and support of emergent domain-specific modelling languages, but there is also a need for tools enabling import/export of learning scenarios *extra*-domains, particularly towards standard abstract EMLs. These import/export facilities can be concretely supported by models transformations (Kurtev, 2005; Mens et al. 2005).

Our current research orientation about MDE theories and practices is based on the assumption that such domains provide a means for supporting these transformation objectives. We have some results of preliminary experiments with some transformations from CPM scenarios (specific to Problem-Based Learning situations) to IMS-LD compliant scenarios (Laforcade, 2005). At this time, we are focusing on the transformation of an abstract scenario into a domain-specific one which will permit:

- the facilitation of the reuse and exchange of learning scenarios (thanks to the global understanding obtained when the scenario is described with the vocabulary of the target community of teachers/designers-community);

- the visual representation, at a knowledge level, of the descriptive scenario of a realized learning situation or the tracks/observations gathered during and after the learning session; in addition, the way in which the descriptive learning scenario and the assessment of the students can be improved.

Knowing that every language is composed of an abstract syntax (the concepts/relations), a concrete notation (the visual formalism) and semantics (Baar, 2006; Muller et al., 2006), we on purpose propose to differentiate *graphical representation* from *domain-specific representation* of an abstract scenario: graphical representation only deals with concrete syntaxes transformations (visualization, binding and abstraction from concrete syntaxes) whereas domain-specific representation has in addition to deal with the mapping of the two different abstract syntaxes (that can be semantically-speaking very 'distant' from each other). In order to reduce the transformations complexity, we chose originally to direct our focus on the graphical representation of abstract scenarios. This enabled us to focus first on the technological obstacles that are related to the abstraction and binding steps we have to identify and overcome before tackling abstract syntaxes transformations.

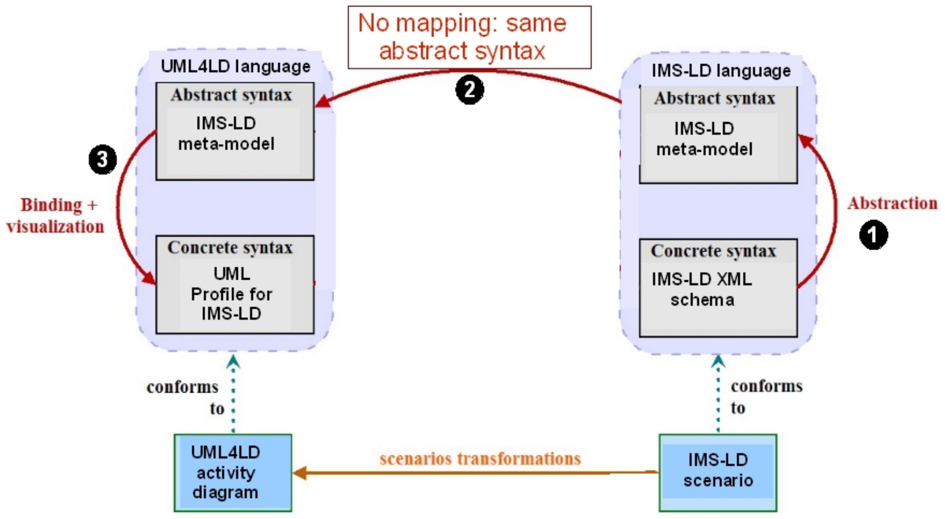

Figure 1: The transformation process from IMS-LD learning scenarios to equivalent UML activity diagrams.

The UML4LD language and tool

For our experiments, we chose the IMS-LD specification as the source abstract modelling language. We chose the UML formalism (OMG, 2003) as the target notation, especially the UML activity diagram representation because this diagram is very often used to visually draw the learning scenarios as a flow of activities between the involved roles. In addition, we chose to exploit the UML extensions mechanisms (via the UML profiles) in order to make the LD-concepts explicit when represented into this graphical notation. We have defined a UML profile dedicated to IMS-LD: the UML4LD profile. This profile is composed of stereotypes and tagged-values defined by extending some UML meta-model elements. We also decided to initially restrict the information gathering from abstract scenarios to particular concepts and relations of IMS-LD, to those that can be easily deduced when looking at an activity diagram as illustrated into the LD Best Practices. Indeed, all IMS-LD concepts cannot be represented into this diagram that is not appropriated to represent objectives, pre-requisites, properties, etc.

All the IMS-LD concepts and relations are represented as a conceptual model (or meta-model). The representation is concretely a UML class diagram. Figure 2 illustrates all the concepts and relations from this meta-model (IMS, 2003a). We have added an area surrounded by a blue line for delimiting the concepts and relations we are interested in. Only these elements will be considered by our transformation process.

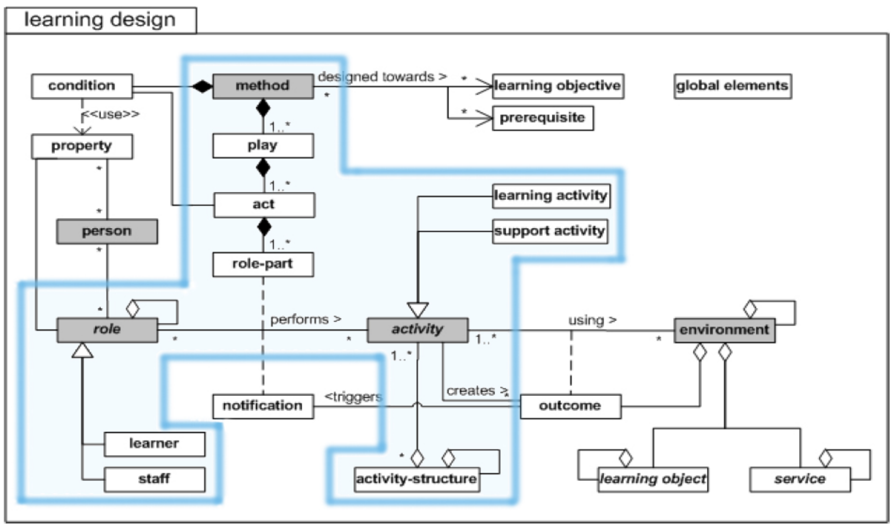

Figure 2: Class diagram of the IMS-LD meta-model showing the IMS-LD concepts and relations, and those concerned by our transformation process.

The UML4LD Profile Elaboration and implementation

Although there is no abstract syntaxes mapping to establish in this experiment, our choice to elaborate a UML profile implies that we have to elicit the meta-model elements from the UML language which will be extended by stereotypes (these last ones mapping with the LD-concepts). Table 1 presents some examples of such mappings.

Table 1: Some mappings between IMS-LD abstract concepts and elements from the UML concrete syntax

learner	Partition	learner
	Actor	learner
staff	Partition	staff
	Actor	staff
learning-activity	Operation	learning-activity
	ActionState	learning-activity
support-activity	Operation	support-activity
	ActionState	support-activity
activity-structure	Operation	activity-structure
	SubActivityState	activity-structure

Another justification of the need of stereotypes creation is illustrated in Table 1: the stereotypes enable the reader to differentiate *staff* role from *learner* roles just by looking at the activity diagram (this approach adds value as it is not currently used with hand-made activity diagrams from the LD best practices (IMS, 2003b)). Figure 4 illustrates a concrete example of this representation. One can notice that an LD-concept is related to several UML model elements stereotyped identically. This choice is justified because of the self-structure of the UML syntax and the tooling we put into practice that forces us to some specific constructs (for example the creation of an *actor* before the creation of a *partition*). For similar reasons, *activities* are mapped to both *ActionState* and *Operation*, except activity-structures that are mapped to *SubActivityStates* (in order to embed other activities).

Other LD-concepts and relations have more complex mappings to UML elements because of their association to UML relations: for example, the *role-part* concept is mapped to the nested containment of an *ActionState* to a *Partition* (via the '*Assignment'* association defined in the UML meta-model) (OMG 2003). Another complex mapping is that related to the sequencing of IMS-LD acts, concretely represented by a complex management of UML *transitions* into the activity diagram.

The UML4LD profile is concretely implemented as a UML profile into the *Objecteering* CASE-tool because of our own experience of this environment (Softeam, 2007) and because it proposes

- an internal and proprietary language, 'J', dedicated to the handling, navigation, and creation of UML models;
- some functional libraries allowing the creation of models elements as well as the creation of representation elements, and other libraries allowing to handle the XML-format of the LD source scenarios.

The transformation process is performed by mean of the selection of the associated service integrated into the contextual menu appearing when a UML package is selected.

The Transformation Process

We chose to sketch the transformation process we have designed because the details are too much related to the specific tool we used. On the other hand, we think that this abstract process can be applied to other tooling. The transformation process is completed in two-steps:

- from an IMS-LD model, selected by the user, a corresponding UML model is created (only UML model elements are generated, not the graphical view); this first step deals with abstraction actions (from the XML-formatted source file) and with binding actions (towards equivalent UML model elements tagged thanks to the UML4LD profile extension elements).
- these UML model elements are then automatically projected: creation of the corresponding representation elements (boxes, links, etc.) into an activity diagram (visualization step).

The first step is concretely realized by an *imperative* model transformation (Mens et al, 2005): the IMS-LD XML file is sequentially interpreted. According to the XML tags encountered, some specific model creation actions are performed in order to generate the UML4LD target-model. The concrete realization of the transformation code allowed us to highlight the complexity of transformation models when dealing with various concrete syntaxes. Even if there is no mapping between abstract syntaxes (the LD concepts and relations are the same with both notations), the transformation process has to deal with binding abstraction (from the XML model) and then with binding realization (towards the UML model). Concretely, these two steps reveal technological obstacles we need to overcome. For example, some IMS-LD concepts are *implicit* into the XML binding: there are no tags for *explicitly* declaring them (eg. the acts sequencing, the collaborative activities, etc.). The second step of the transformation process (creation of the representation elements) is very specific to the tool we chose to use to integrate and develop our prototype. The code still has to be improved with positioning algorithms and with a more complex layout management in order to avoid the overlapping of elements and cross-linking.

Illustration and discussion

The previous process has been implemented with several case studies extracted from the LD-*Best Practices*. (IMS, 2003b). By comparing the generated activity diagram with the one illustrated in the *Best Practices*, the experiment led us to perfect the code transformation. This section focuses on one of these case studies: the "*Problem Based Learning*" scenario.

```
...
    <imsld:learning-design identifier="Problem-Based-Learning" ...>
      <imsld:components>
        <imsld:roles>
          <imsld:learner identifier="R-student"/>
          <imsld:staff identifier="R-facilitator"/>
                    ...
        </imsld:roles>
        <imsld:activities>
          ...
          <imsld:activity-structure identifier="AS-Prepare" structure-type="sequence">
            <imsld:title>Prepare</imsld:title>
            <imsld:learning-activity-ref ref="LA-Read-problem-Desc"/>
            <imsld:learning-activity-ref ref="LA-Choose-Chairperson"/>
          </imsld:activity-structure>
        </imsld:components>
      <imsld:method>
        <imsld:play identifier="PLAY-PBL">
            ...
          <imsld:act>
            <imsld:role-part>
              <imsld:role-ref ref="R-student"/>
              <imsld:activity-structure-ref ref="AS-Prepare"/>
            </imsld:role-part>
            <imsld:role-part>
              <imsld:role-ref ref="R-facilitator"/>
              <imsld:activity-structure-ref ref="AS-Help-Group"/>
            </imsld:role-part>
           </imsld:act>
          ...
        </imsld:play>
        ...
      </imsld:method>
    </imsld:learning-design>
```

Figure 3: Extracts of the abstract scenario of the '*Problem-Based* Learning' case-study illustrated into the IMS-LD Best Practices.

Figure 3 is an extract of the learning scenario specified with the IMS-LD specification for this case-study. One can notice the static specification of the different kinds of roles and activities between the 'Components' section whereas the dynamic of the scenario is expressed between the 'play' tags by means of a sequence of 'act'. Each one of these acts contains at least one 'role-part' that concretely associates one role and one activity to each other. More than one role-part into an act indicates concurrent activities.

Figure 4 is the UML activity diagram proposed in the best practices to graphically illustrate the textual narration for the '*Problem-Based* Learning' case-study. This UML diagram have been probably made directly with a drawing tool allowing UML representations. Contrary to our tool proposition, this activity diagram have not been automatically built.

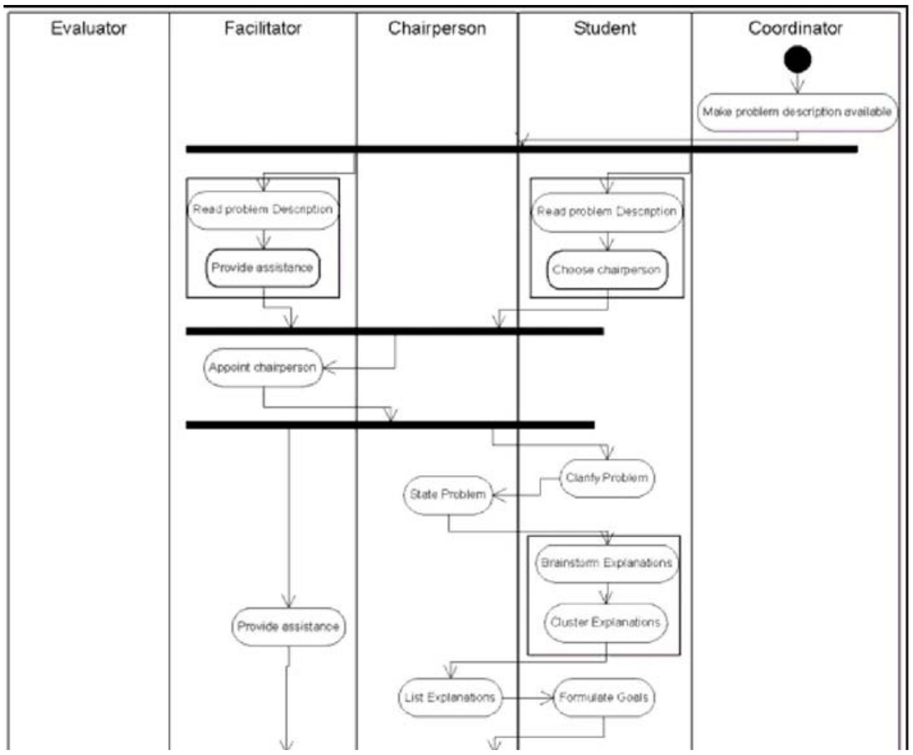

Figure 4: UML Activity Diagram

By applying our transformation process to this case-study, via the tooling we developed, a final activity diagram is generated. Figure 5 is an illustration of the result. This activity diagram has been automatically generated from an IMS-LD XML file. This figure is a good illustration of the value added by the stereotypes in order to make explicit the LD-concepts (the representation also uses automatic coloring, line-styles, etc. in accordance with the IMS-LD concepts). The three kinds of activities can easily be distinguished as well as the various kinds of roles.

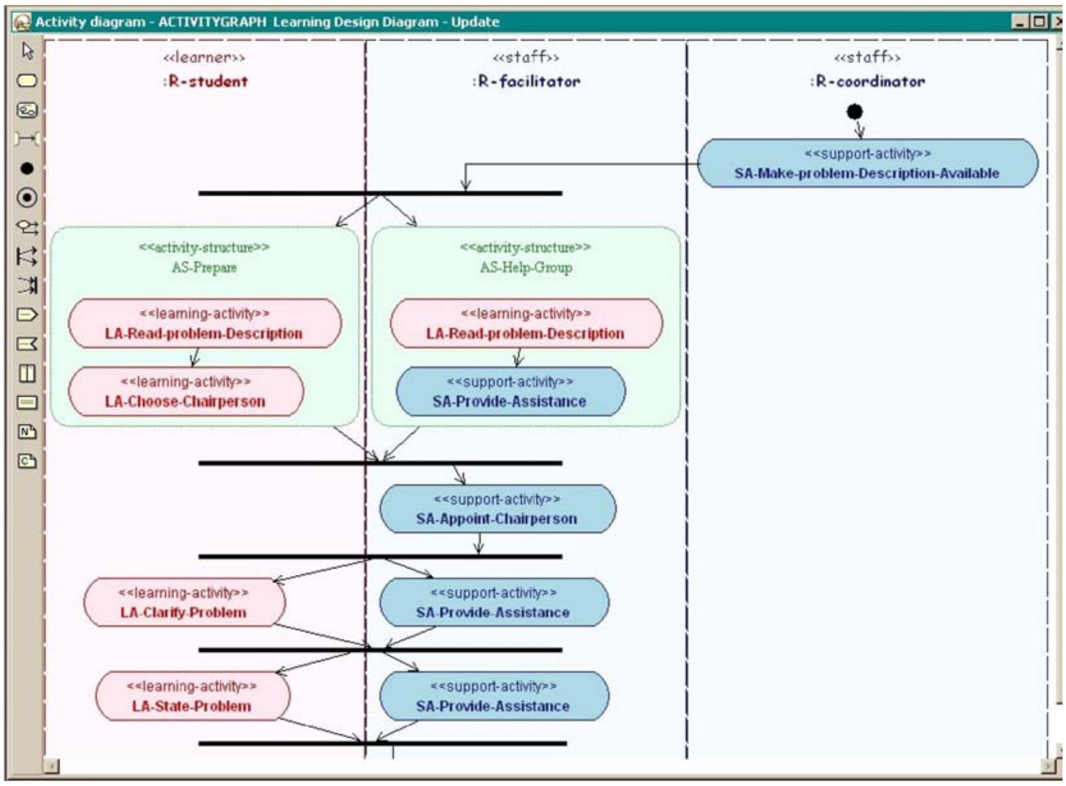

Figure 5: Extract of the UML activity diagram automatically generated by the UML4LD transformation tool

Comparing the two versions, one can notice that the *Provide-assistance* support-activity can be factorized into only one activity that will be performed in parallel with the *Clarify-Problem* and *State-Problem* activities (similarly with the left diagram). This problem appears because several *acts* refer to the same activity in the XML version of the source scenario.

We can also notice that the 'learning-activity' *Read-Problem-Description* is used by a 'staff' role; indeed the IMS-LD meta-model does constrain staff roles to only perform support-activities (see Figure 2). Whatever the designer's intention, this example shows that misconceptions (when so recognized) can be easier to detect with graphical representations; it also illustrates the benefit of the UML4LD tool in assisting designers when reusing/appropriating scenarios from other designers or when re-engineering their own learning scenarios.

We are now working on the improvement of the UML4LD notation: other UML concepts, such as the 'branch' pseudo-states, can be used to visually improve understanding of the scenario. We also plan to experiment more deeply with the LD-*Best Practices* case studies (IMS, 2003b) in order to study the pedagogical expressiveness of the IMS-LD specification. Finally, we expect to complete the UML4LD language and tool by the reciprocal transformation (from activity diagrams to IMS-LD scenarios) in order to propose a round-tripping authoring-tool exploiting UML activity diagrams.

For now, the UML4LD tool graphically visualizes IMS-LD scenarios but does not allow the creation or editing of IMS-LD representations. It can be used by teachers/designers according to the pre-requisite that they have been introduced to the use of the Objecteering CASE-tool on top of which we plug our transformation facility. Even if they do not need to be expert in the use of this proprietary tool, they have to be familiarized with it. This is the main disadvantage of using existent tools. Finally, UML4LD users interact with the tool following the 3 steps process illustrated in Figure 6: 1/selection of the IMS-LD file; 2/automatic generation of UML4LD model elements; 3/ generation of the activity diagram representation by asking the dedicated command from the activity graph element generated on step 2.

Finally, teachers/designers can benefit from the UML4LD tool for

- improving the design of IMS-LD learning scenario: visualizing the in-design scenario helps to highlight misconceptions, to validate the proposed learning-flows, to improve the understanding and involvement of all design actors, to generate some documentation diagrams, etc.

- improving the reuse of IMS-LD learning scenarios by visually helping teachers/designers to understand and appropriate existent scenarios that are exchanged through repositories of IMS-LD XML files, or that have been made by themselves a long time ago, etc.

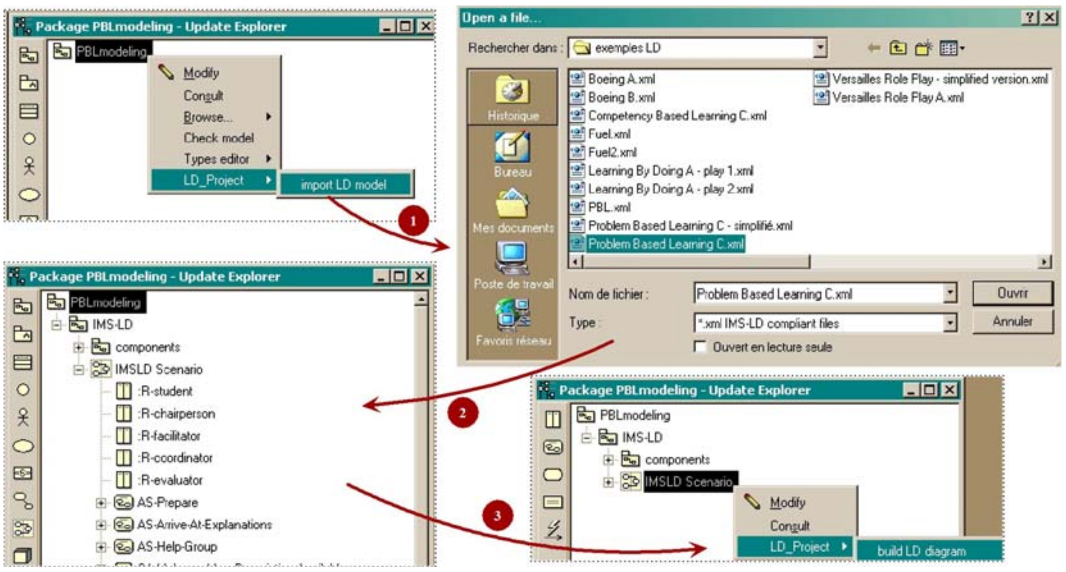
Package PBLmodeling - Update Explorer
PBLmodeling
Modify
Consult
Browse...
Check model
Types editor
LD_Project
import LD model
1
Open a file...
Rechercher dans : exemples LD
Historique
Bureau
Mes documents
Poste de travail
Favoris réseau
Boeing A.xml
Boeing B.xml
Competency Based Learning C.xml
Fuel.xml
Fuel2.xml
Learning By Doing A - play 1.xml
Learning By Doing A - play 2.xml
PBL.xml
Problem Based Learning C - simplifié.xml
Problem Based Learning C.xml
Versailles Role Play - simplified version.xml
Versailles Role Play A.xml
Nom de fichier : Problem Based Learning C.xml
Type : *.xml IMS-LD compliant files
Ouvert en lecture seule
Ouvrir
Annuler
Package PBLmodeling - Update Explorer
PBLmodeling
IMS-LD
components
IMSLD Scenario
:R-student
:R-chairperson
:R-facilitator
:R-coordinator
:R-evaluator
AS-Prepare
AS-Arrive-At-Explanations
AS-Help-Group
2
Package PBLmodeling - Update Explorer
PBLmodeling
IMS-LD
components
IMSLD Scenario
Modify
Consult
LD_Project
build LD diagram
3

Figure 6: The different actions needed from the UML4LD users to generate the activity diagram representation.

Conclusion

This article has presented and discussed a Model Driven Engineering approach applied to the instructional design domain, specifically focussing on models transformations between abstract scenarios and domain-specific ones. We have then presented experimental examples of the graphical representation of IMS-LD scenarios into UML activity diagrams. Even if the initial idea was to simplify scenarios transformations (without taking into account abstract syntaxes mappings that deal with the pedagogical expressiveness of both the source and target educational modelling languages), this research shows that when dealing with concrete syntaxes bindings, many technical and technological problems also arise. For example, it is important to clearly separate representation elements from model elements. We also highlighted the difficulty of abstracting a specific binding, and then of applying another one: the transformation code has to deal with the explicit/implicit expressiveness of both source and target notations.

We think that a formalization effort of current and future educational modelling languages will help the binding abstraction/application works and also anticipate some reflexions about the mappings of different pedagogical modelling languages which is likely to be of great interest in the near future to the instructional design community (mappings will be needed in order to compare, exchange, reuse and assemble learning scenarios produced by various EMLs). Indeed, we also claim that current Model-Driven Engineering principles, techniques and tools will help in supporting the emergence of domain-specific and visual educational modelling languages as well as their bridging.

References

Baar T. (2006). Correctly Defined Concrete Syntax for Visual Modelling Languages. In MoDELS'06, the 9th International Conference on Model-Driven Engineering Languages and Systems. Springer, LNCS 4199, pp. 111-125.

Botturi, L., Stubbs, T. (eds.) (forthcoming in 2007). *Handbook of Visual Languages in Instructional Design: Theories and Practices*. Hershey, PA: Idea Group.

Chikofsky E. J., Cross II J. H. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1).

De Vries, F., Tattersall, C. & Koper, R. (2006). Future developments of IMS Learning Design tooling. *Educational Technology & Society*, 9 (1), 9-12.

Favre, J.M. (2004). Towards a Basic Theory to Model Driven Engineering. In WISME'04, the 3rd Workshop in Software Model Engineering. Lisbon, Portugal.

IMS (2003a). IMS Learning Design Version 1.0 Final Specification. Technical report. Retrieved from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

IMS (2003b). IMS LD Best Practice and Implementation Guide. Technical report. 2003. Retrieved from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html

Kent S. (2002). Model Driven Engineering. In *IFM 2002*, LNCS 2335, pp. 286-298, Springer-Verlag.

Kinshuk, Sampson D.G., Patel A., Oppermann R. (2006). Special issue: Current Research in Learning Design, *Journal of Educational Technology & Society*, V(9)-1.

Koper, R. (2006). Current Research in Learning Design. *Educational Technology & Society*, 9 (1), 13-22.

Koper R., Miao Y. (2006). Using the IMS LD Standard to Describe Learning Designs. Submitted Book Chapter. *EETC*, Open University of the Netherlands.

Kurtev I. (2005). Adaptability of Model Transformations. PhD Thesis, University of Twente, 90-365-2184-X.

Laforcade P. *(2005).* Towards a UML-based Educational Modelling Language. *ICALT'05*, 5–8 July, 2005, Kaohsiung (Taiwan), p. 855-859.

Laforcade P., Choquet C. (2006). Next Step for Educational Modelling Languages: The Model Driven Engineering and Reengineering Approach. *ICALT'06*, pp. 745-747.

Martínez-Ortiz I., Fernández Manjón B., López Moratalla J., Moreno-Ger P. (2005). Towards the Implementation of IMS Learning Design in the LMS. In the proceedings of the International Conference of Web-Based Education WBE'05, Grindelwald, Switzerland. Uskov V. Eds.

Mens T., Van Gorp P. (2005). A Taxonomy of Model Transformation. In the proceedings of the International Workshop on Graph and Model Transformation, Tallinn, Estonia.

Muller P.-A., Fleurey F., Fondement F., Hassenforder M., Schneckenburger R., Gérard S., Jézéquel J.M. (2006). Model-Driven Analysis and Synthesis of Concrete Syntax. In MoDELS 2006, the 9th International Conference on Model-Driven Engineering Languages and Systems. Springer, LNCS 4199, pp. 98-110.

Newell A. (1982). The Knowledge Level. *Artificial Intelligence*, 18 (1).

OMG (2003). UML v1.5. Report formal/03-03-01. Retrieved from http://www.omg.org/technology/documents/vault.htm#modelling

Paquette G. (2004). Educational Modelling Language, From an Instructional Engineering Perspective. Retrieved from http://www.licef.teluq.uquebec.ca/gp/fr/publications/documents/ArticleEML-MISA.doc

Paquette G., Léonard M., Lundgren-Cayrol K., Mihaila S., Gareau D. (2006). Learning Design based on Graphical Knowledge-Modelling. *Educational Technology & Society*, 9 (1), 97-112.

Reload (2007). The official Reload Learning Design Editor web site. Retrieved from http://www.reload.ac.uk/ldeditor.html

Softeam (2007). Objecteering official website. Retrieved from http://www.objecteering.com