Graphical representation of abstract learning scenarios: the UML4LD experimentation

Pierre Laforcade
LIUM / IUT de Laval
52 rue des Docteurs Calmette et Guérin
53020 Laval Cedex 9, France
pierre.laforcade@lium.univ-lemans.fr

Abstract

This communication concerns the presentation of an experimental research work about the graphical representation of abstract learning scenarios. We also on purpose propose some reflexion elements about the stakes of models transformation in order to generate domain-specific representation of abstract scenarios. Our experiment concerns the automatic generation of a UML activity diagram from an IMS-LD learning scenario.

1. Introduction

Present Educational Modeling Languages (EML) [1] are used to formally describe abstract learning scenarios, ensuring by this way reuse, exchange and interoperability over several Learning Management Systems (LMS). The Learning Design specification [2] (IMS-LD) is the current standard for these EML. The resulting LD-scenarios are formatting by the means of a 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 (as advocated in the IMS-LD best practices [3]).

The research work presented in this paper deals with the graphical representation of abstract learning scenarios previously specified with an EML. We aim to provide teachers/designers with dedicated tools that can visually represent abstract scenarios. These tools will then help designers to understand the scenarios, insuring by this way their reuse and their exchange.

Our work takes place into a Model-Driven-Engineering (MDE) context [4]. We explain in [5] the interests and stakes of 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 then in 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 transformation of models between abstract scenarios (specified with an EML) and domain-specific scenarios (human-readable), ii) presenting the technical process that we have experimented in order to demonstrate the add-value of a graphical representation, automatically generated, of an abstract learning scenario.

2. Research context

Our current concern is only focusing on the design phase (not the deployment/runtime phase) of learning scenarios. We define a domain-specific scenario as a learning scenario described in a human-readable way addressing teachers/designers. It generally uses a visual formalism (textual as well as graphical). This scenario aims at describing explicitly the mental representation of the learning situation during design, at a knowledge level [6]. 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 of reflexion. A domain-specific modelling language can be more or less formal, and more or less operational (i.e. having a XML binding). The vocabulary (concepts and relations) is the one of the teachers/designers community. Its pedagogical expressiveness is generally specific to a pedagogical approach or other learning specificities. CPM [7], MISA/MOT+ [8] are examples of such languages.
On the contrary, an abstract scenario is firstly specified and formatting into a machine-readable/interpretable way. Indeed, 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 the 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 intrinsic objectives). IMS-LD is an example of EML.

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 support by models transformations [9]. Transformations from domain-specific scenarios to abstract ones aim at 'abilities' support (reuse by wider communities, interoperability on various LMS, etc. On the contrary, reverse transformations aim at specific-designers appropriation.

Knowing that every language is composed of an abstract syntax (the concepts/relations), a concrete notation (the visual formalism) and semantics, we on purpose propose to differentiate graphical representation from domain-specific representation of an abstract scenario: the first one only deals with concrete syntaxes transformations whereas the second one has also in addition to deal with the mapping of the two different abstract syntaxes.

In order to reduce the transformations complexity, we chose to firstly conduct our works on the graphical representation of abstract scenarios. So, we can focus on the technological obstacles we have to overcome before tackling abstract syntaxes transformations.

3. Experimentation: 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 [15] as the target notation, especially the UML activity diagram representation.

In addition, we chose to exploit the UML extensions mechanisms (via the UML profiles) in order to make possible the explicitation of LD-concepts when represented into this graphical notation. Concretely, we have so defined a UML profile dedicated to IMS-LD: the UML4LD profile, concretely implemented as a UML profile into the Objecteering CASE-tool.

This profile is composed of stereotypes and tagged-values defined by extending UML meta-model elements. The information gathering from abstract scenarios is restricted to the IMS-LD concepts and relations that can be easily deduced when looking at an activity diagram (as illustrated into the LD Best Practices: learner-role, staff-role, learning-activity, support-activity, activity-structure, play, act, role-part, etc.).

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 to map with the LD-concepts. The table 1 presents some examples of such mappings.

<table>
<thead>
<tr>
<th>IMS-LD concept</th>
<th>UML modelling element</th>
<th>Stereotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>learner/staff</td>
<td>Partition</td>
<td>learner/staff</td>
</tr>
<tr>
<td>learning-activity/support-activity</td>
<td>Operation</td>
<td>learning-activity/support-activity</td>
</tr>
<tr>
<td>activity-structure</td>
<td>Operation</td>
<td>activity-structure</td>
</tr>
<tr>
<td>SubActivityState</td>
<td>ActivityState</td>
<td>activity-structure</td>
</tr>
</tbody>
</table>

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

Thanks to the stereotypes the reader of a generated activity diagram will be able to differentiate staff role from learner ones just by looking at the graphical representation: this point is a great add-value that is not currently present in the activity diagram from the LD best practices (see figure 1).

A LD-concept is related to several UML model elements that are stereotyped identically because of a/ the self-structure of the UML syntax; b/ the tooling we put into practice that forces us to use some specific constructs. For similar reasons, activities are mapped to both ActionState and Operation, except activity-structures that are mapped to SubActivityStates (in order to contain 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. Another complex mapping is the one related to the sequencing of IMS-LD acts, concretely represented by a complex management of UML transitions into the activity diagram.

The transformation process is completed in two-steps: 1/from an IMS-LD model, selected by the user, a corresponding UML model is created, 2/ these UML model elements are automatically projected as representation elements (boxes, links, etc.) into an activity diagram. The first step is an imperative model transformation: the IMS-LD XML file is sequentially interpreted. According to the XML tags encountered, some specific model creation actions are performed.

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 tag for explicitly declare them (eg. the acts sequencing, the collaborative activities, etc.).

The second step of the transformation process (creation of the representation element) is very specific to the tool we chose to use; so we do not detail it.

The previous process have been concretely experimented on several case studies extracted from the LD-Best Practices. By comparing the generated activity diagram with the one illustrated in the Best Practices, the experiment led us to perfect the code transformation.

The figure 1 is an illustration of our result from the “Problem Based Learning” case study: the diagram is obtained thanks to our transformation from the XML code (taken from the LD Best Practices).

4. Conclusion

This article has presented and discussed, in a Model Driven Engineering approach applied to the instructional design domain, our interests about models transformations between abstract scenarios and domain-specific ones. We have then presented, in a second time, our experimentation works about the graphical representation of IMS-LD scenarios into UML activity diagrams. These works show that dealing with concrete syntaxes bindings also comes with many technical and technological problems: the transformation code has to deal with the explicit/implicit expressiveness of notations.

We are now working on the improvement of the UML4LD notation and we also plan to experiment more deeply the LD-Best Practices case studies in order to study the pedagogical expressiveness of the IMS-LD specification.

5. References