Specification of learning management system-centered graphical instructional design languages

A DSM experimentation about the Moodle platform

Esteban Loiseau[1], Pierre Laforcade[1]
[1]*Laboratoire d'Informatique de l'Université du Maine, Avenue Olivier Messiaen, Le Mans, France*
{esteban.loiseau, pierre.laforcade}@univ-lemans.fr

Keywords: Domain-Specific Modeling, Meta-modelling, instructional design, learning management system

Abstract: This paper presents a 6-month explorative research work about the specification of specific instructional design languages. These languages have to tackle a double objective: capturing teachers-designers' needs and practices and guarantee a model in conformance with an existent Learning Management System. Domain Specific Modeling techniques are used to both specify these languages and to provide practitioners with some graphical authoring-tools. This explorative research work has been conducted in relation with a pedagogical engineering team, specialized in Moodle, from Le Mans University. Three different DSM approaches have been experimented and analyzed. This research is part of the French ANR GraphiT Project.

1 INTRODUCTION

Many institutions provide teachers and students with some Learning Management Systems (LMS). These educational platforms are not restricted to online uses. They are also useful when combined with face-to-face teaching for blended learning. Teachers can use them for providing students with some materials or to support the some complex collaborative learning situations involving a strong use of the platform communication features. In order to set up such complex activities, teachers must develop some designer skills about how and when managing and sequencing the available features and tools. Such skills can be acquired through specific education programs generally focusing on the technical aspects of the platform. They are rarely about designing learning situations on the considered LMS. Because of the multiplicity of educational theories and approaches, as well as the lack of tools and processes dedicated to existent LMSs, teachers develop some *ad-hoc* and individual learning design techniques. In such contexts, it seems relevant to help teachers in focusing on the design for the specific LMS they have at their disposal. A focus on the instructional design possibilities and how they can rely on the platform features should encourage individual reflection about the design of learning situations.

The GraphiT project (funded by the French Research Agency) is based on an LMS-centered designing approach. Within this starting project, we have conducted a 6-months exploratory research. Its main objective was to investigate some Domain Specific Modeling (DSM) techniques for helping the specification of LMS-centered graphical instructional design languages as well as the development of dedicated editors. This paper focuses on its presentation and on the analysis of its results.

2 RESEARCH CONTEXT

2.1 The GraphiT Project

This project (Graphit, 2013) has started in February 2012. Its main goal is to study the possibilities and limits about the pedagogical expressiveness of *operationalizable* languages to specify: future leaning scenarios could be fully deployed and automatically operated on an existent LMS. Such instructional design languages aim at promoting and improving the uses of current LMSs by providing practitioners with some LMS-specific

designing language and authoring-tool. To achieve these goals, the Graphit project is based on three assumptions: 1/ LMSs implicitly embed their own *instructional design paradigm* (vocabulary, rules, constraints, etc.); 2/ it is possible to make these *instructional design domains* explicit; 3/ their identification and formalization allow to build and to develop external LMS-centered tools. We also assume that *instructional design domains* are stable enough (through versions, extensions, integrated external tools) to guarantee the durability of the instructional design artifacts that will be specified and developed.

The project scientific approach is illustrated in Figure 1. For every considered LMS, we propose: a/ to make explicit the platform instructional design domain; b/ to extend the learning platform with an adaptative import/export API; c/ to make explicit the teachers' designing needs & practices for this platform; d/ to specify and support some LMS-centered and practitioners-directed instructional design languages and editors.

The main challenge of this project is to abstract enough the LMS instructional design to propose teachers some higher design blocks. The LMS expressiveness and limits have to be overcome in order to offer teachers some instructional design mechanisms closer to their practices and needs for specifying and sequencing the learning activities to perform. Our idea is to conduct the platform abstraction in accordance with the formalization of future learning scenarios. This LMS-centered approach guarantees that learning scenarios could be operationalized into the LMS. The underlying scientific issue relies on the identification of the inter-relations between pedagogical expressiveness and operationalization support.

The identification of practitioners' needs (Figure 1 - left) and the formalization of the LMS instructional design domain (Figure 1 - right) are the prerequisites and first blocks for the GraphiT project. They have already provided some results (Clayer, Toffolon and Choquet, 2012; Abedmouleh and Laforcade, 2012). The project main issue and objective (Figure 1 - center) has been investigated through the initial exploratory works, explained in this paper.

2.2 Targeted instructional design languages

The project aims at proposing tools and instructional design languages in conformance with these properties (1) *graphical formalization* (specific visual formalism for representing the learning scenarios) and (2) *operationalization ability* (output scenarios are machine-readable through the API to integrate to existent LMSs - cf. Figure 1).

The second property relates to the *Educational Modeling Languages* (EML) and their *binding* concept (Koper and Manderveld, 2004). Such EMLs often aim at being generic: their educational expressiveness is *independent* from Technology-Enhanced Learning systems like LMSs, and *neutral* about the instructional design practices they cover. EMLs focus on the scenarios formalization and *executability* towards LMSs. The experiments about the extension of the MOODLE LMS for importing learning scenarios conformed to IMS-LD (the EML *de facto* standard), proved that adapting existent LMSs requires some complex and heavy re-engineering (specific runtime-engine to integrate) in order to overcome the limits of the platform features (Burgos, et al. 2007). EMLs fail to provide a support for operationalizing EML-conformed learning scenarios into existent LMSs.

The graphical property fits more to VIDLs (Visual Instructional Design Language) (Botturi and Stubbs, 2008). These languages offer some visual notations from simple drawing with a few symbols to complex diagrams. VIDLs focus on higher-level languages, i.e. with syntaxes and semantics closer to some instructional theories or to some specific communities of instructional designers' practices.

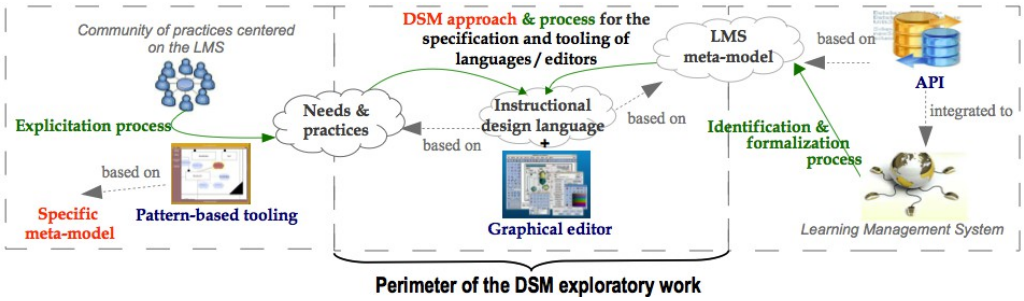

Figure 1: Targeted contributions of the Graphit project (tools in blue, processes/methods in green, models/techniques in orange, specific results in grey) and perimeter of the exploratory work.

VIDLs aim at supporting imagination, creative thinking, communication, etc. Because VIDLs are rather visual domain-specific languages focusing on human-interpretations, they do not systematically provide some binding techniques. Learning models are generally saved as proprietary files mixing learning scenarios data with graphical information. Some VIDLs or authoring-tools provide instructional designers with a standardised EML binding by the means of a dedicated export service (Dodero, Martinez del Val and Torres, 2010). Nevertheless, resulting models are generally less expressive than the original scenarios because of the semantic gap between the source/target languages.

One can consider the instructional design languages targeted by the GraphiT project as a mix between VIDL (because of the graphical formalization) and EML (because of the binding to the LMS explicit instructional design domain). Within the project the operationalization will be realized thanks to a dedicated adaptative import/ export API (presented in (Abedmouleh and Laforcade, 2012)). According to the *elaboration* criteria for VIDLs classification (Botturi, et al. 2006), the GraphiT project languages will be designed at *implemen-tation* (LMS-centered design) and *specification* (design directed towards teachers' needs and practices) levels. The main issue relies on studying the limits of the visual expressiveness, in accordance to teachers' practices, while constraining the operationalization on a specific LMS.

2.3 Domain Specific Modeling

Domain Specific Modeling (DSM) (Kelly and Tolvanen, 2008) is a software engineering methodology involving the systematic use of domain-specific languages to represent the various facets of a system. They are specific to a domain and can be defined as the set of concepts and their relations within a specialized problem field. They offer primitives whose semantics are familiar to all practitioners in that domain (in opposition to generic-purpose languages like UML). This methodology aims at reducing the software engineering costs by automating the generation of the application source code and by allowing the handling/execution of the produced models. Potential challenges and issues regarding the application of DSM techniques and tools within the instructional design domain have been discussed in (Laforcade, 2010).

The Graphit project uses the DSM principles as a *methodological* framework to specify languages and as a *practical* framework to guide the development of the related graphical editors. It provides a very challenging trend for supporting the specification of human-interpretable visual models with machine-readable persistence.

We conducted a very first experiment about the Moodle LMS (Abedmouleh and Laforcade, 2012). We apply the DSM techniques and tools to specify a graphical language and editor on top of the Moodle metamodel (produced through the identification process): semantics are voluntary limited to the LMS expressivity in terms of instructional design abilities. This experimentation succeeded in tackling the binding and operatio-nalizing objectives. Nevertheless the added value of such external authoring-tools is limited because of the semantics restriction: the learning situations designed are too close to the LMS semantics.

2.4 Objectives of the exploratory work

This paper focuses on a further exploratory research work. Its main objective is to explore several DSM approaches to specify instructional design languages at a first abstraction level from the LMS-specific expressivity (Figure 1 center). In order to achieve this goal, we chose a specific LMS and defined a first need in terms of pedagogical practices. The LMS metamodel, capturing its instructional design abilities, as well as the operationalization API, are considered as known and functional (obtained from (Abedmouleh et al., 2012)). In order to encompass the metamodeling subjectivity, our results will focus on concepts, relations and properties of the LMS but not on their syntactic representation formalized by the metamodeling technique.

3 REQUIREMENTS ANALYSIS

3.1 The Moodle platform

Moodle is currently deployed and available for blended learning at Le Mans University. It is a widespread educative platform (Moodle, 2013), largely used by public institutions. Moodle is supported by a large, still growing, users community. It is an open-source, modular, easy to extend PHP Web application. Moodle was designed with a social constructivist approach in mind (Dougiamas and Taylor, 2003). It provides several features (resources and activities) like collaborative tools and services (forum, chat, wiki and others).

3.2 Pedagogical Activities

In order to identify some very first needs to capture, we worked with a pedagogical engineering team, named PRN, from Le Mans University. Their missions include managing the deployed Moodle *instances* and training teachers. Recently, the PRN has trained teachers about instructional design aspects. They also take in charge the manual operationalization of learning scenarios designed by some learning teachers-designers involved in distant learning programs. The PRN engineers are skilled and experimented Moodle users. We considered them as appropriate interlocutors considering our exploratory research work. With their approval, they gave us an access to their teachers training materials and to some relevant online courses. The analysis of these materials let us identify a first abstract need from the platform's features. From a single tool, for example a forum, one can design several pedagogical uses, depending on its configuration. We then compiled a non-exhaustive list of pedagogical activities, from these concrete sources and from other uses found in the domain literature (Conole et al., 2004). In order to complete these future elements for the language to specify, we also added some VIDLs recurring structural elements (selection, sequence, conditional activities, etc.).

3.3 Manual Binding

We first manually match the pedagogical needs and the Moodle features in order to later formalize within the instructional design language. Assessment activities such as *self-assessment*, *summative assessment* or *formative assessment* rely on the same Moodle *quiz* feature but on different settings of its parameters, including answering modalities (examples : number of attempts, "adaptative mode"). Some pedagogical activities can be set up on the platform in a variety of ways. According to the users choices while designing a scenario, the equivalent properties/values will drive the elicitation of the most appropriated LMS feature to use. For example, a *debate activity* can be conducted synchronously with a *chat*, or asynchronously with a *forum.* The *Writing a report activity* is a more complex example, with three properties to value (collaborative versus individual modality, online versus offline, and iterative or finalist writing). Four different Moodle tools can be chosen (with proper settings in the 8 cases) : *online text assignment*, *wiki*, *blog* or *file submission*. To implement the structural elements of a pedagogical scenario, we have to exceed the Moodle limitations making uncommon uses of Moodle features: there are no activity-structures and conditional branching in the 1.9 version used at Le Mans University. To implement *sequence* and *choice* activity-structures we used *labels* to provide students with instructions. Moodle does not provide conditional activities before the 2.3 version: it is possible to use *groups* and *groupings* to limit availability of an activity to a particular group of students and to assign each groups to an alternative activity. Nevertheless, branching conditions cannot be automatically checked: a *label*, not visible by students, can be used as a reminder for the teacher about assigning students to groups.

4 IDENTIFYING THREE APPROACHES

According to the DSM definitions, the graphical instructional design languages we propose to specify are specific modeling languages. Every modeling languages can be defined as a tuple *<AS, CS*, M*ac, SD, Mas>* (Chen et al., 2005) where AS is the abstract syntax, CS* are the concrete syntaxes, M*ac is the set of mappings between abstract syntax and concrete syntaxes, SD is the semantic domain, and Mas is the mapping between abstract syntax and semantic domain. The abstract syntax (AS) defines, in a structural way, the concepts and relations of a modeling language. It is concretely formalised with a metamodel. This metamodel also specifies the future conformed models (*binding*).

Our instructional design languages have to meet two requirements: providing a human- interpretable graphical notation (a specific graphical concrete syntax), and also providing a machine-readable notation (serialisation format) for the scenarios conformed to the Moodle metamodel, in order to be handled by our Moodle import/export API. A first approach consists in defining the abstract syntax as the exact Moodle metamodel (considered pre-existent for this exploratory research work) in order to achieve, first and foremost, the machine-readable requirement. The graphical concrete syntax will be derived from the Moodle metamodel. Nevertheless, this approach has to tackle the expressivity limits of models when only based on the graphical notation expressiveness. The second approach is about extending the abstract syntax, initially the Moodle metamodel, with new syntactical elements adding the required semantic (pedagogical activities and

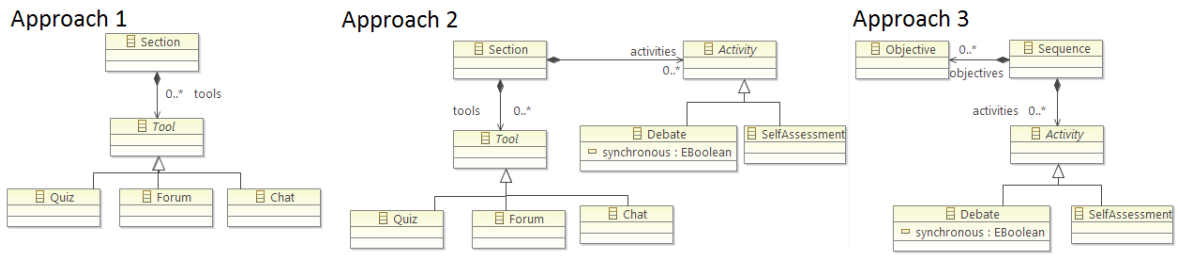

Figure 2: Partial representations of the abstract syntax according to the three approaches

structural elements) non-covered by the initial abstract syntax. By adding new elements to the abstract syntax, we break the conformity to the initial metamodel for the future models. This approach must address such issue within the design-time (i.e. a post-design processing is not relevant in this approach), in order to study the DSM tooling capabilities. The third approach is opposed to the first one, focusing on a language firstly capturing the teachers-designers' needs and practices: the abstract syntax is defined with no relations with the Moodle metamodel. This approach will produce non-conformed models that cannot be directly operationalized in Moodle. Such approach will have to provide some model transformations in order to restore the platform-conformance.

5 TOOLING EXPERIMENTS

5.1 DSM Tooling

We chose to use additional frameworks from the open-source project (Eclipse, 2013). EMF allows to define the abstract syntax (metamodel) and to drive the business code generation, GMF to specify the graphical modeling languages and to generate the editor code, ATL to specify and to execute some transformation rules. The GMF framework provides the functionalities to define modeling languages according to the DSM methodology. Every syntax and mappings are models, modifiable through an Eclipse integrated editor: the abstract syntax (domain model) is a metamodel conforming to the *Ecore* meta-metamodel; concrete syntax can be defined through two models: *gmfgraph*, defining the graphical notation, and *gmftool*, specifying the concepts and relations available in the diagram editor toolbar. Abstract/concrete syntax mapping (Mac) is described through the *gmfmap* model. The semantic domain and its mapping towards the abstract syntax are not explicitly defined in their own specific models, but can be specified through OCL rules in addition to the domain model. Two additional models are involved in the editor code generation: *genmodel* model, precising the code generation parameters (for example the models persistence format) and *gmfgen* model, parameters about the editor code generation. The three approaches we propose are based on an initial metamodel formalizing the Moodle instructional design domain (Abedmouleh et al., 2012).

5.2 Tooling the first approach

This approach is about specifying concrete syntax of the language through the *gmfgraph* and *gmftool* models, using the Moodle metamodel as the abstract syntax (figure 2 left). Figures, icons, labels, properties, etc. representing the toolbar and the graphical elements can be specified to hide the underlying platform concepts, in order to provide a more significant representation (pedagogical in our particular case). On one hand this approach is appropriate for situations such as the *assessment* activities, when a single platform tool is used (1-1 *mapping*), with different pre-configured properties. Indeed, GMF provides an initialisation feature, allowing instantiation of several domain concepts triggered when creating an element. On the other hand it is an issue when some specific information are required to identify the right platform feature. GMF cannot automatically triggers some specific initialisation actions when the properties of an existent instance are dynamically modified. In order to overcome this difficulty, we have to define one tooling element for each combination of settings of a pedagogical activity in order to maintain the same level of expressiveness. For example *Writing a report activity* should exist in 8 different variants, according to the 3 criteria values, available in the editor toolbar.

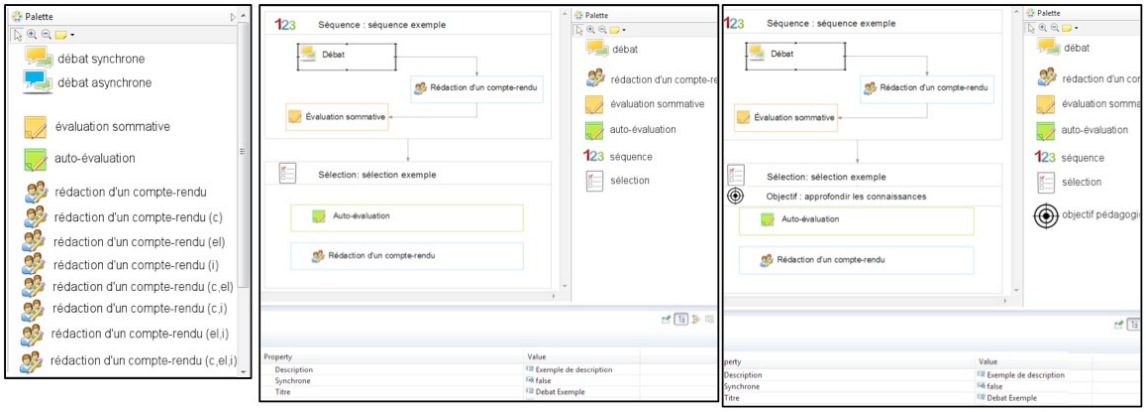

Figure 3: Partial editors screenshots of editors obtained through (approach 1 on the left, approach 2 in the middle and approach 3 on the right).

5.3 Tooling the second approach

Some techniques of the first approach can be reuse to process the simple binding cases. For the others, the domain model have to be extended with new concepts (for example the *debate* pedagogical activity with the *synchronous* property - cf. figure 2 center) and mapped with graphical representations.

Unfortunately, the GMF framework relies on this metamodel to drive the persistence of the future models created with the graphical editor. Extending the initial metamodel breaks the conformance of future models with the dedicated operationalization API. However, we can address this issue by mixing meta-modeling techniques (for example we use the *transient* property in order to disable the persistence mechanism) and code modification (for example if the *synchronous* property of a *debate* activity is set to *true*, a *chat* element is created and the corresponding *forum* element, if previously created, is removed).

5.4 Tooling the third approach

This approach relies on the formalization of a metamodel specific to teachers' needs and practices presented in 3.2. This new metamodel specifies some relevant pedagogical activities (non-exhaustive), the structural/sequencing elements and allows the definition of pedagogical objectives (figure 2 right). These *objectives* were on purpose added because this approach does not focus, in the first place, on the binding towards any LMS conformance. The specification is only limited by the teachers' needs and our ability to formalize them by using metamodeling techniques. The GMF processes we conducted are the same as for the other two approaches: definition of the graphical notation, of the toolbar, etc.

The models produced with such DSM generated editor do not comply with the initial Moodle metamodel, thus the learning scenarios cannot be operationalized on Moodle. Because the Moodle metamodel is not *included* in this specific metamodel, we cannot reuse the techniques (metamodeling plus code modification) from the second approach to restore a Moodle conformance. The use of model transformations, through ATL rules for example, is a potential solution but it requires an additional step after the learning scenario design and before the operationalization. Models produced with GMF-based editors are saved in an XML/XMI format, enabling also some XML-oriented transformation (with XSL for example). This transformations must be defined and specified in accordance with an educational engineer and can be very complex to produce. Some instructional design concepts could not have a matching set of elements in the LMS language (incomplete mappings lead to inconsistent scenarios), or, on the contrary, source elements may not be enough detailed to make a transformation decision. For example in our case study, the *pedagogical objectives* have no matching concepts in the Moodle language; thus, they have to be skipped in the operationalization phase (semantic losses). Such ignored elements, if important for the scenario, can make other elements inconsistence.

6 DISCUSSION

We analysed and compared the 3 DSM approaches with four criteria: 1/ visual expressiveness (the

semantic expressiveness from models as perceived by the teachers when manipulating visual elements); 2/ abstract expressiveness (the real model semantics, according to underlying abstract syntax, not directly perceived by teachers); 3/ the potential of operationalization for the produced models; 4/ the operated scenario semantics (i.e. the semantics of the resulting course setting up on the LMS during the operationalization step). The first criteria does not require some specific computer-science skills and is considered as an end-user point-of-view. We gathered feedbacks, collected during an informal meeting, from the involved PRN members. Table 1 summarises the results, marked +, / or – according to the two languages requirements of this study (pedagogical expressiveness and LMS binding).

From a graphical expressiveness perspective, approaches 2 and 3 are more relevant because they do not require end-users to anticipate the most appropriate detailed concepts at the design-time when picking up elements from the toolbar. For example in the approach 1, the teacher-designer has to choose between *Synchronous debate* and *Asynchronous debate* activities before adding the activity to the drawing space; a later change of mind will require to delete the previous chosen activity from the diagram and to pick up / instantiate the other one. In opposition, approaches 2 and 3 only require for designers to choose the most relevant activity from the toolbar, and to decide its properties later. The third approach does not deal with the LMS binding or conformance at first (when design scenarios). Instructional design languages from this approach provide some elements, properties and relations directly addressing the teachers' needs. There are no constraints or limits like in the second approach focusing on extending the LMS metamodel.

The abstract expressiveness of the produced models is directly related to the initial choices defined in accordance with the three approaches. The compliance of the produced models is straightforward for the first approach, while it still requires some adaptations in the other two ones. In the second approach we used metamodeling techniques and code modifications to maintain the models persistency in conformance with the LMS metamodel. For the third approach, a more complex binding is required after the design-time (with models transformation for example). Our experiment showed that such transformations can easily become more complex and time-consuming. Also, this approach cannot guarantee the LMS conformance for the produced scenarios. Previous researches on model transformations between practitioners-oriented learning scenarios and LMSs-specific instructional design metamodels already revealed such issue (Nodenot et al., 2008; Abdallah, Toffolon and Warin, 2008). In opposition, the second approach maintains the initial semantics by using both metamodeling techniques and compliance restoring techniques ; it is strongly dependent on the weaving used to extend the initial platform metamodel. The first approach guarantee the conformance of the semantics by directly using the initial metamodel.

The third approach corresponds to the usual way to specify a VIDL with its main advantage (expressiveness) but also inconvenience (difficulty to operationalize). The first approach reveals the limits of the concrete syntax expressiveness when only defined by derivation of the abstract syntax: this approach can map several representations with a single concept or relation from the LMS metamodel but this relation of derivation is immutable and cannot be dynamically changed (DSM limit) to improve the user-friendliness. The second approach is intermediate on all criteria: best expressiveness / LMS compliance ratio. However, it requires a strong metamodeling expertise to reduce the developing cost while restoring the LMS compliance. This approach highlights the importance to drive the expressiveness (and semantics) extension of the initial metamodel with the binding capacity.

Criteria	Approach 1	Approach 2	Approach 3
Visual expressiveness	Too many elements (-)	Elements provided are coherent with teachers' needs but constraint (/)	Elements provided are coherent with teachers' needs (+)
Abstract expressiveness	Limited to the LMS one (-)	Limited to a close perimeter of the LMS one (/)	Not limited (+)
LMS metamodel conformance	Direct (+)	Requires technical fine-grained adjustments during the design-time (/)	Requires complex coarse-grained transformations after the design-time (-)
Scenario semantics after import on the LMS	Maintain (+)	Maintain but constraint (/)	Equivalent course / scenario can be inconsistent/incomplete (-)

Table 1: Comparison of the three approaches

Matching the teachers' needs and practices to the LMS features cannot reduce to a programming task. It has to be made explicit by involving teachers (validation of matching rules and constraints) and DSM experts. Such explicit informations can be used as a base for the formalization of the metamodel extensions.

7 CONCLUSION

This paper presents a six-month exploratory research. Its main objective was to study several Domain Specific Modeling approaches to specify/ develop an instructional design language/editor specific to an existent Learning Management System. The targeted language had to meet two requirements: expressiveness directed towards teachers-designers' needs and practices, and binding/persistence centered on the platform metamodel. For the study we choose Moodle as the LMS to comply with, and we restricted the teachers' needs to some pedagogical activities and usual activity-structures. From the DSM theoretical framework we identified three approaches to put into practice with the DSM tooling from the Eclipse Modeling Project. The three results have been analyzed and compared. The approach offering the best compromise for both requirements consisted in extending the initial platform metamodel in order to include the first abstraction level from the platform features. This research was part of the starting Graphit project. Further researches are currently conducted, still meeting the same two requirements than from this first study but with a scope expanded to several LMSs and to more complex teachers-designers' needs and practices. The Domain Specific Modeling methodology, as well as Model Driven Engineering techniques, will be deeply studied, in particular the models weaving techniques. Indeed, such techniques allow to explicit the mappings relations between teachers' practices and platform features. Such explicit and formal relations could help us in identifying a specific way to extend the LMS semantics while maintaining the models binding to the LMS metamodel.

ACKNOWLEDGEMENTS

This work is funded by the French GraphiT project [ANR-2011-SI02-011] (http://www-lium.univ-lemans.fr/~laforcad/graphit/).

REFERENCES

Abdallah, F., Toffolon, C., Warin, B., 2008. Models transformation to implement a PBCL Scenario : Moodle case study. *IEEE ICALT 2008.*

Abedmouleh A., Laforcade P., 2012. Improving the design of courses thanks to graphical and external dedicated languages: a Moodle experimentation, *1st Moodle Research Conference,* Heraklion.

Abedmouleh A., Oubahssi L., Laforcade P., Choquet C., 2012. An analysis process for identifying and formalizing LMS instructional language, *ICSOFT 2012*, Rome.

Botturi L., Stubbs T., 2008. *Handbook of Visual Languages in Instructional Design: Theories and Pratices*. Hershey, Information Science Reference.

Botturi L., Derntl M., Boot E., Figl K.A, 2006. Classification framework for educational modeling languages in instructional design, *IEEE ICALT 2006*, Kerkrade.

Burgos D., Tattersall C., Dougiamas M., Vogten H., Koper R., 2007. Mapping IMS Learning Design and Moodle. A first understanding, *Journal of universal computer science*, 13(7).

Chen K., Sztipanovits J., Abdelwalhed S., Jackson E., Semantic Anchoring with Model Transformations, *ECMDA-FA 2005*, Nürnberg.

Clayer J-P., Toffolon C., Choquet C., 2012. A pattern based and teacher-centered approach for learning design, *CATE 2012*, Napoli.

Conole G., Dyke M., Oliver M., Seale J., 2004. Mapping pedagogy and tools for effective learning design, *Computers & Educations*, 43(1-2), pp.17-33.

Dodero J., Martinez del Val A., Torres Jorge, 2010. An extensible approach to visually editing adaptive learning activities and designs based on services, *Journal of Visual Languages & Computing*, 21(6).

Dougiamas M., Taylor P.C, 2003. Moodle: Using Learning Communities to Create an Open Source Course Management System, *EDMEDIA*, Honolulu.

Kelly S., Tolvanen J-P., 2008. *Domain-Specific Modeling:Enabling Full Code Generation,* Wiley-IEEE Society Press.

Koper R., Manderveld J., 2004. Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning, *British journal of educational technology*, 35(5), pp.537-551

Laforcade P., 2010. A DSM approach for supporting the specification of Visual Instructional Design Languages and the building of dedicated editors, *Journal of Visual Languages & Computing*, 21(6), pp.347–358

Nodenot T., Caron P-A., Lepallec X., Laforcade P., 2008. Applying Model Driven Engineering Techniques and Tools to the Planets Game Learning Scenario, *Journal of Interactive Media in Education*, 23.