Making explicit the Moodle instructional design language

Abstract— **The use of existent LMSs presents many difficulties related to the design and operationalization of learning scenarios. Teachers have to encompass the LMS technical features and services in order to understand the underlying way of designing. Generic instructional design editors fail in bridging the gap between how they design a learning scenario and how the learning session can be set up within the target LMS. If LMSs could be able to explicit their intrinsic and implicit learning design model, it can be exploited as a proprietary format to build tools and facilities dedicated to this LMS. The research presented in this paper aims to present our method in terms of necessary analysis and steps for the identification and the formalization of Moodle instructional design language. The method takes into account three different viewpoints: a viewpoint centered on the LMS macro-HMIs (Human-Machine Interfaces), a functional viewpoint and a micro viewpoint. We validate the proposed method by applying this formalization on two versions of Moodle.**

Keywords - Learning Management System; E-learning; Instructional Design; Operationalization; Technology Enhanced Learning; Process; Moodle

I. INTRODUCTION

Many universities have adopted web-based LMSs as the TEL system. They use them to offer teachers a range of pedagogical and administrative tools for supporting teaching and learning activities [1]. However, many teachers have difficulty to create learning designs that are truly engaging to their students [2]. They are not familiar with the implicit learning design domains of LMSs [3]. Most of open source LMSs are very difficult to apply in real schools, because they require some effort to be appropriated [4].

Due to the complexity of LMS functionalities, users are expected to have some pre-existing knowledge of these functionalities. Despite online forums, it is still difficult for a teacher to design his courses on platforms. LMSs are in continuous evolutions and discussions regarding different versions of a platform are interwoven. Many forums have input from developers, and software architects, which is hardly understandable for non-LMS-experts. In addition, there is no support (nor human nor software products) able to help teachers in clarifying, defining and then specifying their learning situations before setting-up them within the LMS. They have to appropriate the various screens and form-based interfaces to abstract some low-level details to think about their global design courses.

Teachers need solutions to narrow the gap between their educational intention and the pedagogical features proposed by the LMS at their disposal. They ask for appropriate tools helping them in understand the underlying "way of thinking and designing" of this LMS. In our work, we aim at supporting practitioners to overcome these LMSs' obstacles in order to help them in focusing on the design of learning situations. Our contribution consists in extracting, identifying, and formalizing the LMS implicit instructional design language. We also on purpose propose a meta-model formalism to capture it. The meta-model is obtained by the abstraction of pedagogical features and services provided by the considered LMS. This meta-model acts, according to the language theory, as an abstract syntax. It will then be used as a basis for the development of external editors [5] [6].

In [7], we have presented this meta-model-based approach for identifying and formalizing LMS languages. In this paper, we apply our approach on Moodle (section 2). Section 3 is dedicated to the comparison of two different versions of Moodle according to their meta-models. Section 4 concludes our paper and presents our perspectives.

II. MOODLE 2.4 CASE STUDY

In this section we present the application of the identification and formalization process on Moodle 2.4 [7] for many reasons: (1) Moodle is increasingly used in schools, universities and companies, (2) Moodle is also used in our university, and (3) Moodle has an active community that continuously develops APIs and tools. Note that the version 2.4 is the installed version in our university.

A. Application of the Macro-HMI Analysis on Moodle

The application of macro-HMI analysis on Moodle consists in identifying interfaces related to course design. We analyzed interfaces titles and navigation paths / URLs.

We studiously browse all the links in a specific interface. These links often point to new interfaces. Moodle is designed based on a top-down approach: the main interface is about specification and presentation of the course content, other interfaces (like add a forum, a label...) are accessible from the main interface.

The figure 1 shows the result of applying the macro-HMI analysis on Moodle. A course is composed of categorie(s), outcome(s), scale(s), section(s), group(s), grouping(s) and one question bank. Course sections are organized into resources and activities for students. Moodle 2.4 offers 7 resources (Book, Page, Label, IMS content package, File, Folder, and URL) and 13 activities (Forum, Database, Glossary, Assignment, Lesson, Quiz, Workshop, SCORM package, External tool, Choice, Survey, Wiki, and Feedback). In figure 2, we present only one resource (Label), and 5 activities (Survey, Chat, Workshop, Quiz, and Forum) for clarity reasons.

In the page specification of each concept, attributes are divided into different parts. For example, for the Chat activity, its fields are divided into 4 parts named: general, common module settings, restrict access, and activity completion. These parts names are presented in the macro-HMI model. Note that there are only two types of relationships within this model: composition relationship and inheritance relationship.

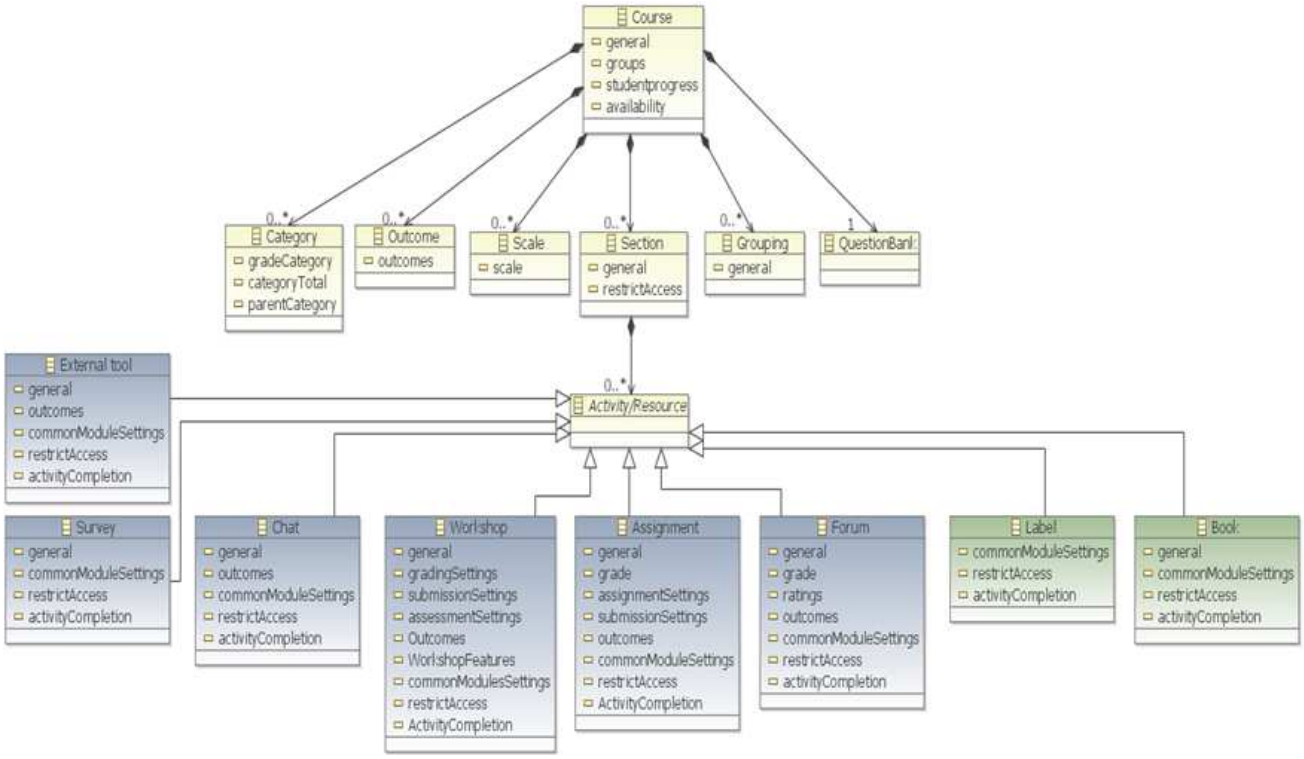

Figure 1. An extract of the Moodle macro-HMI model.

B. *Application of the factorization on Moodle*

We then applied the factorization process. We noticed that all activities/resources had the common attributes: “commonModuleSettings”, “restrictAccess”, and “activityCompletion”. So we moved these attributes to the Activity/Resource class. All activities had the common attribute “general” according to the macro-HMI model, that’s why we created a class called “Activity” and we moved the attribute “general” into it. Some Moodle activities could have outcomes like Chat activity, Workshop, and Quiz. We added in the macro model a class named “ActivityWithOutcomes”. This class had “outcomes” as an attribute. We noticed that some activities with outcomes could be graded. Therefore, we added the class “GradedActivityWithOutcomes”.

Among “GradedActivityWithOutcomes” class, some activities had the common attributes “grade”. The “ActivityWithGradedSection” class is created and contained the “grade” attribute. Some activities from the “ActivityWithGradeSection” had the common attributes “ratings”. The class “ActivityWithRatingsSection” is added to the macro model with the attribute “ratings”. All coming steps are carried out on the basis of this analysis.

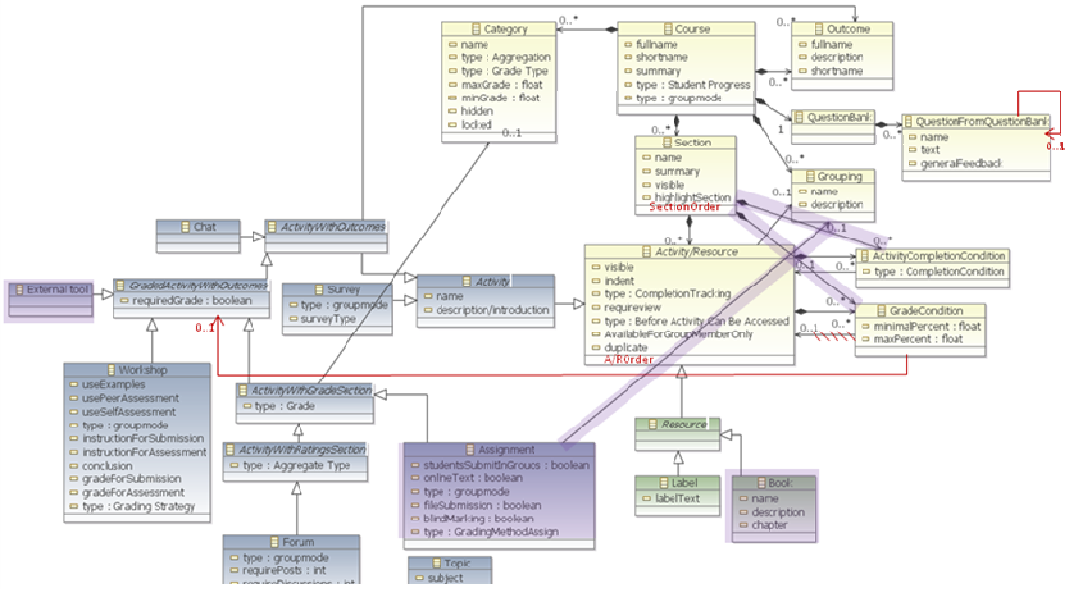

Figure 2. An extract of the Moodle micro-HMI model (without corrections in red), and an extract of the Moodle 2.4 final model (with corrections in red).

C. *Application of the functional analysis on Moodle*

Based on the macro-HMI model, we proceeded to the functional analysis on Moodle. We divided each interface to several areas. Then, for each area, we studied the graphical interface components to identify functionalities related to instructional design. For example, from the main interface of a Moodle course, a teacher can show/hide/move a section. He can modify the course description, and manage different groups. He can also add an activity/resource in a specific section. If the teacher adds a forum, he will be pointed to a new page about forum specification. He can add files, add/modify/delete/separate a discussion and also reply to a discussion.

We have grounded the formalism of the functional model on the SADT (Structured Analysis and Design Technique) Model [8]. SADT is a multi language supporting the communication between users and designers. It is based on simple concepts in an easy graphical and textual formalism. This language is conformed to our functional analysis approach: top-down, hierarchical, modular and structured. This analysis is very important in our process; it can verify existence and relation between macro-HMI elements.

D. *Application of the micro analysis on Moodle*

The micro analysis consists in several sub-analysis: the micro-HMI analysis, the technical analysis, and the confrontation and formalization process.

1) *Micro-IHM analysis*

The application of IHM-micro analysis is about characteristics identification of instructional design elements. It is based on the macro and functional models.

For example, the "Course" class has "general" as attribute. In this phase, we study in details fields with pedagogical use related to this attribute. "Fullname" and "shortname" are these fields, so we replace "general" attribute in the macro-HMI model by "fullname" and "shortname" attributes in the micro-HMI model.

The figure 3 shows an extract of Moodle micro-HMI model (without taking into account corrections in red). Reference relationships appear in this model. For example the abstract class "ActivityWithOutcomes" refers to "Outcome" class: a teacher can define outcomes to a course then he can associate a specific outcome to Moodle activities except for Choice, Survey, Wiki and Feedback activities.

2) *Technical analysis*

The technical analysis consists in analyzing the Moodle database. Our goal is to identify the Moodle instruction design language from a technical viewpoint to approve the relevant of specific data for this language.

This analysis consists in specifying the reduced Conceptual Data Model for Moodle in relation with the instruction design. We have reviewed all Moodle database tables. Titles semantic analysis of tables and fields allows to (1) gather the tables related to the ID, and (2) ignores those related to technical specifications (users' management, learners' tracking…). Then we studied dependences and relations between database tables. The generated Conceptual Data Model is based on reverse engineering rules. Foreign keys enable the specification of required multiplicities.

3) *Confrontation and formalization*

The micro-HMI analysis and the technical analysis have specified two Moodle instructional design models according to two different viewpoints. In this step, we are interested in the confrontation of these models to formalize Moodle instructional design language. This step is very important in our process. We think that the use of only one analysis method presents many negative points. For example, the micro-HMI model depends directly on the Moodle analyst competence. This means the possibility lack of pedagogical attributes. Similarly, the technical analysis is not an easy task. Many data structures are not explicitly reported when creating the database.

From the 2 models comparison, we noticed that every element/ attribute existing in the micro-HMI model is certainly presented in the technical model. But some elements exist in the technical model without being present in the micro-HMI model. That is why we refer to the PHP source code analysis of Moodle to verify the presence of these elements.

Figure 3 (including corrections in red) shows an extract of Moodle final model. Corrections in red present the confrontation result of the two models. For example thanks to the technical analysis, we found that every section has an order. This attribute has not been detected by the micro-HMI model. The code source analysis confirms the presence of this attribute. The attribute "SectionOrder" is presented in the final model.

Figure 3. An example about relationship verification between the "GradeCondition" class and the "Activity/Resource" class.

The confrontation phase allows also rectifying information on the micro-HMI model. Figure 4 shows an example about relationship verification between the "GradeCondition" class and the "Activity/Resource" class.

Based on the micro-HMI analysis, the "GradeCondition" class refers to the abstract class "Activity/Resource" while the same class refers to a graded activity in the technical model. The code source analysis of Moodle conditionlib.php file confirms that the grade condition refers to a graded activity. That is why the reference relationship is between the two classes "GradeCondition" and "GradedActivityWithOutcomes" in the final model. The final model resulting from the confrontation phase formalizes the Moodle instructional design language.

III. COMPARISON BETWEEN MOODLE 2.0 AND MOODLE 2.4 META-MODELS

In this section, we apply our identifying and formalizing approach on Moodle 2.0 then we compare the two meta-models (Moodle 2.0 and 2.4) in order to identify differences between these versions. Figure 4 (including corrections in red) shows an extract of the Moodle 2.0 final model. Table 1 presents the differences between the two meta-models process by being capable of identifying the new functionalities added by Moodle 2.4 developers in comparison to Moodle 2.0 (The whole Moodle's meta-models are available at the following link : http://www-lium.univ-lemans.fr/~laforcad/graphit/wp-content/uploads/2015/02/metamodels.pdf).

TABLE I. DIFFENCE BETWEEN MOODLE 2.4 AND MOODLE 2.0 META-MODELS

	Moodle 2.4	Moodle 2.0	Comments
External Tool class	Yes	No	The external tool activity module enables students to interact with learning resources and activities on other web sites.
Book class	Yes	No	The book module enables a teacher to create a multi-page resource in a book-like format, with chapters and subchapters.
Relation between Section & ActivityCompletionCondition classes	Yes	No	This relation determines any activity completion conditions which must be met in order to access the section.
Relation between Section and GradeCondition classes	Yes	No	This relation determines any grade conditions which must be met in order to access the activity.
Assignment class	Yes	Yes	In Moodle 2.4 we have 1 class for assignment (Assignment) while in Moodle 2.0 we have 4 classes for assignment (Online text, Advanced uploading files, Offline activity and Upload single file).
blindMarking attribute for Assignment class	Yes	No	Blind marking hides the identity of students to markers.
gradingMethodAssignment attribute for Assignment class	Yes	No	This attribute defines the advanced grading method (Simple direct grading, Marking guide, Rubric) used for calculating grades in the assignment.
Relation between Assignment and Grouping classes	Yes	No	Students are able to collaborate on an assignment.

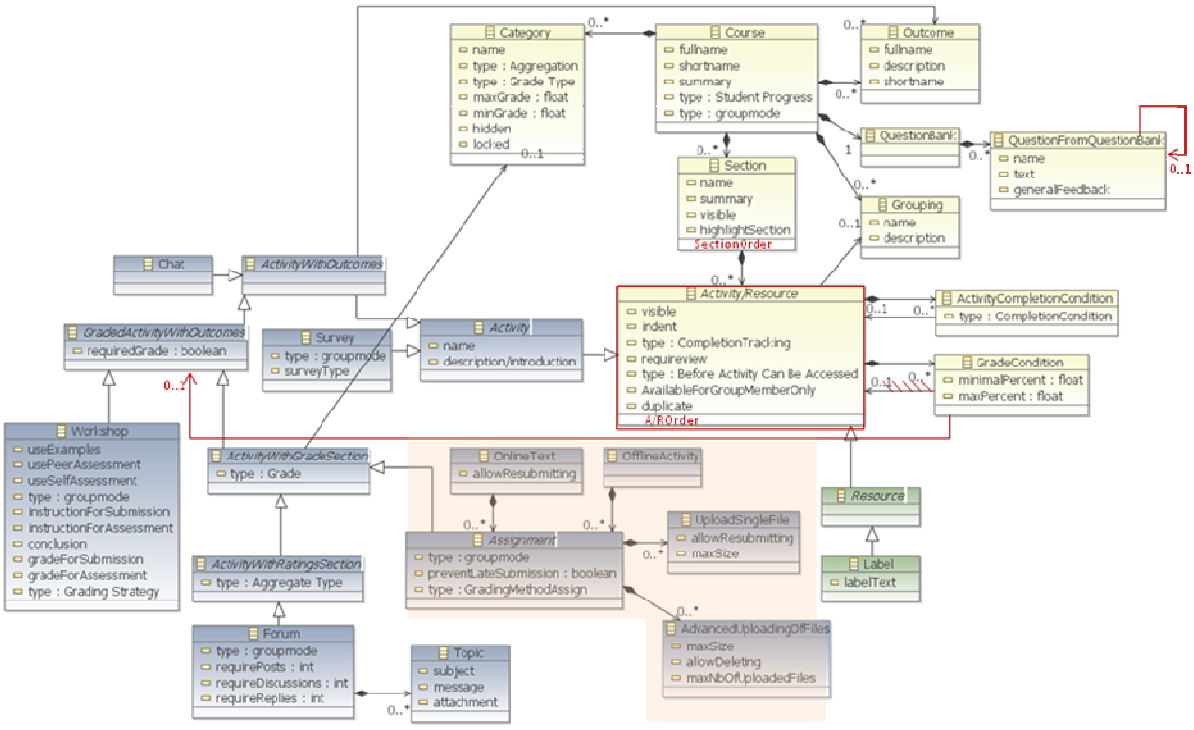

Figure 4. An extract of the Moodle micro-HMI model/ the Moodle 2.0 final model (without/with corrections in red).

IV. CONCLUSION

In this paper, we present the application of a meta-model-based approach and method for identifying and formalizing Moodle 2.0 and 2.4 instructional languages. The comparison between these two meta-models led us to provide a first validation of our process. These meta-models will be used as a basis for the development of the external editor by using a Model Driven Engineering tooling like EMF-GMF. They will guide and generate most of the final code for the editor. We have then been able to propose a graphical and external editor communicating with the system thanks to a dedicated API developed and integrated to the LMS. It will offer more user-friendly and soundly computer artifacts when development is freed from the technological choices related to the initial design of the TEL system considered. This will facilitate the use of LMSs and allow to teachers and pedagogical engineers of becoming more familiar with the specific design upon this LMS. Through the final model, we can confirm that every LMS is not pedagogically neutral but embeds an implicit instructional design language relying on specific paradigms and educative theories followed by the LMS providers. We are applying this method, for validation purposes, on three other LMSs: Ganesha, Dokeos and Sakai.

Our research work allows to teachers-designers designing their entire courses, outside platforms, basing on their pedagogical needs without technical difficulties. Our approach promotes the use of all LMS activities and resources and expands LMS pedagogical concepts not by adding new concepts to users but by facilitating and clarifying the existent tools thanks to the external editor.

REFERENCES

[1] Coates, H., James, R. & Baldwin, G.: A critical examination of the effects of learning management systems on university teaching and learning. Tertiary Education and Management, 11, (2005) 19-36.

[2] Steel, C.: Reconciling university teacher beliefs to create learning designs for LMS environments. In: Australasian Journal of Educational Technology, vol. 25(3), (2009) 399-420.

[3] Martinez-Ortiz, I., Sierra, J.L., and Fernández-Manjón, B.: Enhancing IMS LD Units of Learning Comprehension. In the 4th ICIW, Venice, Italy (May 2009) 561-566.

[4] Mekpiroona O., Tammarattananonta P., Buasrounga N., Apitiwongmanita N., Pravalpruka B. et Supnithia T.: SCORM in Open Source LMS : A case study of LEARNSQUARE. In ICCE2008, Taipei, Taiwan, (2008) 166-170.

[5] Loiseau E, Laforcade P.: Specification of learning management system-centred graphical instructional design languages - A DSM experimentation about the Moodle platform. In ICSOFT'13, Reykjavik, Iceland (2013) 29-31.

[6] Laforcade P. et Abedmouleh A. (2012). Improving the design of courses thanks to graphical and external dedicated languages: a Moodle experimentation. In: Moodle Research Conference 2012, Heraklion, Greece, 14-15 septembre 2012, p. 94-101.

[7] Nour El Mawas Lahcen Oubahssi Pierre Laforcade Aymen Abedmouleh . Towards the identification and formalization of LMS instructional design languages. ECTEL 2014, Graz (Austria); 16-19 sept 2014.

[8] Dao, M., Huchard, M., Rouane Hacene, M., Roume, C., Valtchev, P.: Improving Generalization Level in UML Models Iterative Cross Generalization in Practice, ICCS'04: 12h International Conference on Conceptual Structures, Huntsville, USA (2004) 346-360.