THE IMPLEMENTATION OF THE IMS LD E-COURSE GENERATOR

Goran Savić, Milan Segedinac, Zora Konjović
{savicg,milansegedinac, ftn_zora}@uns.ac.rs
Fakultet tehničkih nauka Novi Sad

Abstract – *This paper presents the implementation of an e-course generator. Courses are generated based on three components: learning goals, learning objects and instructional design. The system's architecture is extensible - it is possible to extend it in order to generate courses in different output format. The result is an automatically generated IMS LD compliant course. The system is implemented in Java programming language.*

1. INTRODUCTION

Creating an e-course may be a very time consuming job if it is done completely manually. Also, there is a need to periodically change the created course. A part of this job may be automated using course generator systems. These systems automatically create an e-learning course using different input parameters – learning objects, domain knowledge, student model, pedagogical knowledge, etc. The paper [1] presents a system DCG which generates a sequence of learning activities using domain knowledge, instructional plan, learning material and student's pre-knowledge as the input parameters. This sequence is personalized for each student. Domain knowledge is represented using a concept map. A learning environment for teaching mathematics is presented in [2]. For each student, the system generates a personalized course. The input parameters are learning objectives, learning material (represented by semantically annotated XML documents), student's profile (knowledge and preferences), teaching scenario (there are 6 predefined global pedagogical strategies) and pedagogical rules (sequencing and selection of learning objects defined using if-then rules). The paper [3] presents the system for automatic generation of a set of individualized hypermedia documents. A course is generated using student's model, domain knowledge, learning objects and a set of rules defined in JESS rule engine. Student's model is also used in [4] where a sequence of learning objects is created using planning mechanism and PDDL plan. This language is used in [5] for course generation, too. Learning activity is chosen in the real-time depending of a learning objective and student's profile. The paper [6] presents the system for the automatic generation of a learning path using ontology of learning objectives and student's model. A similar approach is used in [7], but without the formal representation of learning objectives. A course is generated directly using learning objects and relations are directly defined among learning objects.

This paper presents our system for automatic generation of e-learning courses. As we can see above, all similar systems have two basic components: learning objects and domain knowledge. Our system contains these two components, too. In addition, our system is focused on the instructional design in the course. The system contains a component which formally specifies instructional design used in the course. By changing the instructional design definition, different pedagogical strategies may be easily applied in the course. Then, these pedagogical strategies may be evaluated in order to find the most appropriate strategy for the course. This is the current purpose of our system. Currently, our system doesn't have a student's model as a component and it doesn't create a personalized course.

2. SYSTEM ARCHITECTURE

In this paper, an e-learning course is modelled using the model presented in [8]. The model is based on classical Tyler's rationale [9] and it specifies four modules in a curriculum: learning objectives, learning objects, instructional design and testing strategy. In this paper we are concerned about first three modules. E-courses (in popular e-learning systems or in globally adopted e-learning standards) mostly contain these modules. But, the course is represented as a monolithic unit and the modules are not explicitly represented as separated units. Using such representation, it is not possible to independently change only one module, which is practical demand very often.

Hence, we have decided to represent each module as a distinct component. That way, each component may be independently changed and the components are the input parameters for the automatic generation of an e-learning course. Our system generates a course using three input parameters: learning objectives (goals), learning objects and instructional design. Learning objectives may be defined as learning outcomes that a learner has to achieve. In this paper, we have decided to use ontology for representing learning goals which is an approach used in [4] and [6]. Our ontology of learning objectives is represented in OWL [10] language. In this paper we use the term "learning object" to refer to any digital content that helps a student to achieve a learning objective or evaluates if a learning objective is achieved. Since our system is intended for generating an e-course for web environment, we need learning objects represented in a browser-readable format. So, we have chosen HTML format for learning objects, packed in IMS Content Packaging [11] format. As mentioned, the usage of a learning object is always related to a specific learning objective (for learning or evaluation). Therefore, we created a mapping input component that maps learning objects to learning objectives. We use an XML document to specify this mapping. For generating a sequence of learning activities, it is necessary to define an instructional design used in the course. We have created a

special-purpose XML-based language for describing the instructional design. The language specifies the sequencing and selection of learning objects. An XML schema for this language is presented in [8]. The result of our system is a formal description of an e-learning course. It may be defined in various formats, and we have chosen IMS Learning Design [12] format. A detailed description of all components may be found in [8]. The global architecture of our course generator is depicted in Figure 1.

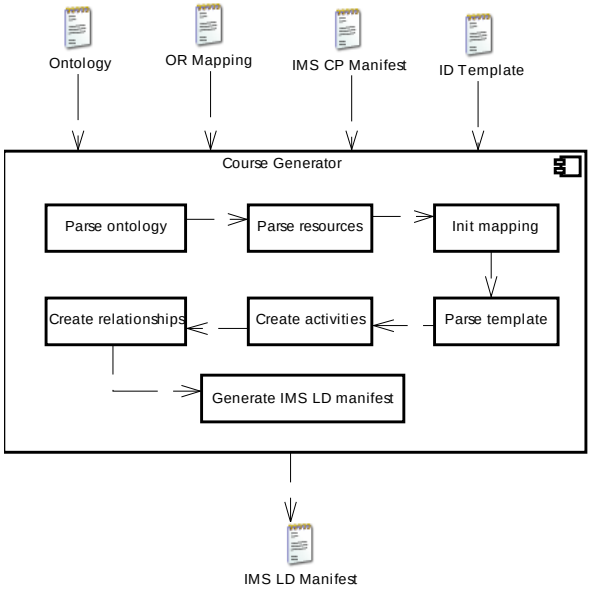

Figure 1. System architecture

3. SYSTEM IMPLEMENTATION

The system is implemented in Java programming language. The system parses the input files and creates the in-memory representation of data read from the input files. Firstly, the ontology of learning objectives and learning objects specification (in the IMS CP manifest file) are parsed. Then, learning objects are linked with learning objectives by parsing the mapping component. The last input file that is parsed is the XML document that describes course instructional design. Using all these data, a sequence of learning activities is generated. Finally, this sequence is a base for generating an IMS LD manifest file, which represents a formal description of our e-learning course. The object models of each component are described below.

3.1. E-learning course

An e-learning course has been modelled using an abstract class `CourseModel`. The course contains learning objectives, learning objects and information about the instructional design. These attributes are initialized from the system inputs. `TemplateParser` class parses the input XML document that describes instructional design. Using initialized attributes, learning activities are created. The course (in the concrete output format) is generated from the sequence of learning activities. `IMSLDCourseModel` class represents the course in the IMS LD format. The methods `createActivities` and `createActivity-Relationships` create the IMS LD representation of learning activities (see section 3.4). The overridden method `generateCourse` creates an IMS LD manifest file. The e-learning course model is shown in Figure 2.

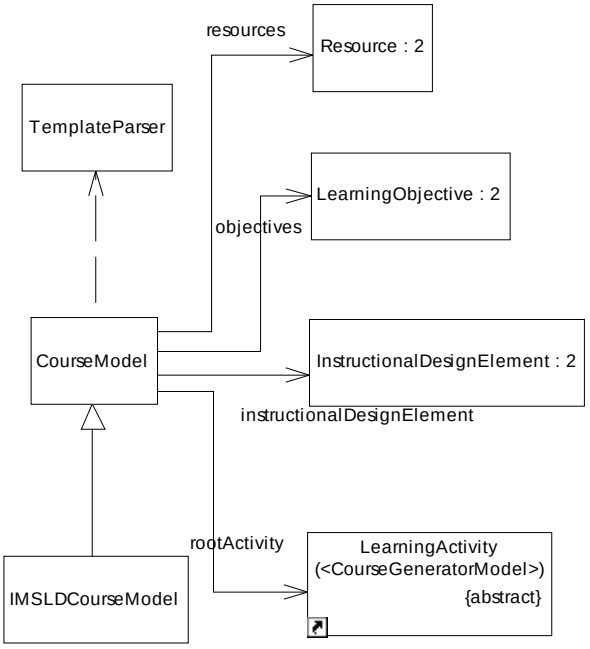

Figure 2. E-learning course model

3.2. Learning objectives and learning objects

For in-memory representation of learning objectives and learning objects, we have created the model shown in Figure 3. `Resource` class represents a learning object. For each learning object, the name, id and file path are managed. In addition, this class has a list of metadata that closely describe the learning object. For a learning objective (class `LearningObjective`), the system specifies the name and hierarchical level. Since learning objectives are hierarchically organized, each learning objective has a reference to its parent learning objective (attribute `parentObjective` in the figure). Likewise, a learning objective has a list of its children. Our ontology defines a relation `surmises` which specifies that a learning objective may be a precondition for other learning objectives. This relation among learning objectives is modelled using `ObjectivePrecondition` class. The attribute `source` in this class represents a learning objective that is a precondition for the learning objective defined by the `destination` attribute. `LearningObjective` class has the list of all `ObjectivePrecondition` objects where that learning objective is the source. Similarly, it contains the list of `ObjectivePrecondition` objects where it is the destination. Mapping of learning objects to learning objectives is modelled in `ObjectiveResource` class.

The class contains a reference to a learning object and to an appropriate learning objective.

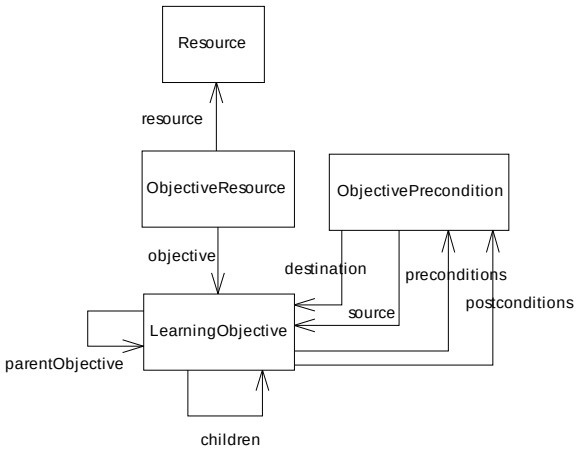

Figure 3. The model of learning objectives and learning objects

3.3. Instructional design

Based on the XML scheme for describing the instructional design in the course, we have created a corresponding object model shown in Figure 4. `InstructionalDesignElement` is a container class and it describes the organization of learning elements in the course. `LearningElement` is a generic learning element in the organization. Learning elements are hierarchically organized, so each learning element contains a reference to its parent learning element and a collection of child learning elements. Also, generic `LearningElement` has a unique identifier represented with `elementId` attribute. There are three different types of learning elements – element group, sequence and learning object. Element group is just a container for other learning elements. Sequence represents a chain of other elements. Its role is similar to the role of loop statements in programming languages. For each sequence, the system iterates through the elements specified with `element` attribute. Learning object is a learning element on the lowest hierarchical level. It actually represents a concrete learning resource. For sequences and learning objects, we should define a specific strategy for selecting learning resources. This strategy is defined in `SelectionRule` element. `SelectionRule` aggregates two lists of `ObjectSelection` elements. First list contains objects that will be included in the course. The second one is for excluded objects. Object selection element specifies learning objects for including or excluding. Learning objects selection is done by specifying the values of its metadata attributes. Also, using the `priority` attribute we specify the order of included learning objects. Learning object may be used for evaluating student's knowledge. In such case, grading information is modelled using `Grading` class.

Beside the course structure, sometimes it is necessary to define relationships between learning elements. For example, in mastery learning student can't proceed to the next lesson if he hasn't completed the previous one. So, we need to define the relationship between two lessons. These relationships are modelled using `Element-Relationship` class. Learning elements which participate in the relationship are modelled using `ConditionElement` class. Learning elements creates the relationship only when a specific condition (modelled with `ElementJoin` class) is satisfied. `Element-Relationship` class contains the `ifCondition` reference which represents a condition for performing specific actions in the course.

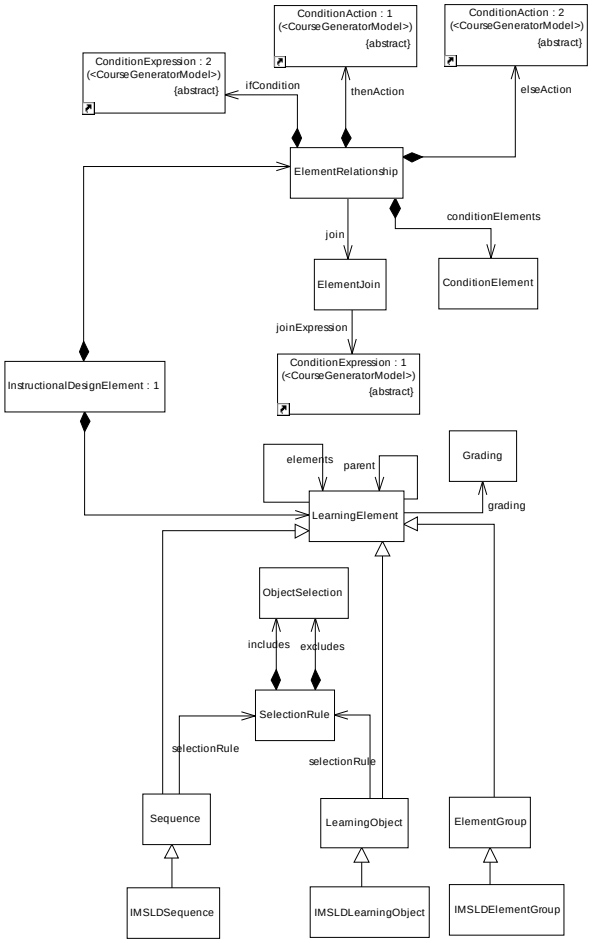

Figure 4. Instructional design model

Conditions are modelled using `ConditionExpression` class. When the condition is satisfied, specific actions defined in the `thenAction` list are performed. Otherwise, actions defined in the `elseAction` list are performed. Actions are modelled using `ConditionAction` class (see below).

`ConditionExpression` is an abstract class that represents a certain condition in the learning process. Condition may contain sub-conditions, which is represented with `childExpressions` attribute. `ConditionExpression` class has successors that represent concrete logical expressions (and, or, equals...). The successors override the method `calculate` that

evaluates the value of the logical expression (true or false). `ConditionExpression` class and its successors are shown in Figure 5.

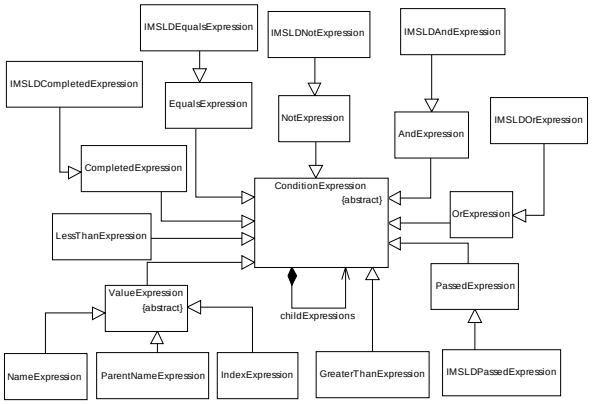

Figure 5. The model of logical expressions in the instructional design description

`ConditionAction` is an abstract class that represents a certain action that should be done in the learning process. Two types of action are supported: showing a learning element (`ActionShow` class) and hiding a learning element (`ActionHide` class). `ConditionAction` class and its successors are shown in Figure 6.

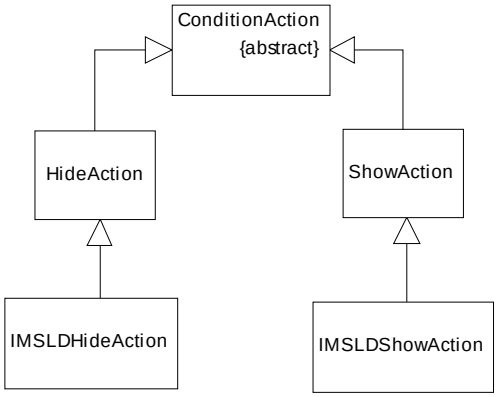

Figure 6. The model of course actions in the instructional design description

3.4. Learning activities

On the basis of input parameters our system creates learning activities in the course. The model of learning activities is shown in Figure 7. An abstract learning activity is represented with `LearningActivity` class. Activities are hierarchically organized. Each activity has a reference to its parent activity (`parentActivity` attribute) and a list of its child activities (`activities` attribute). Grading is a distinct activity and therefore represented with `GradingActivity` class. Grading is always related to a standard learning activity (`learningActivity` attribute), e.g. after the test (standard activity), the teacher gives the grades (grading activity). The relations among learning activities are modelled using `ActivityRelationship` class. The relationship contains a list of activities that form the relation (`learningActivities` attribute). When the `ifCondition` attribute is satisfied, the course performs the actions defined in the `thenAction` attribute. Otherwise, the actions specified in the `elseAction` attribute are performed.

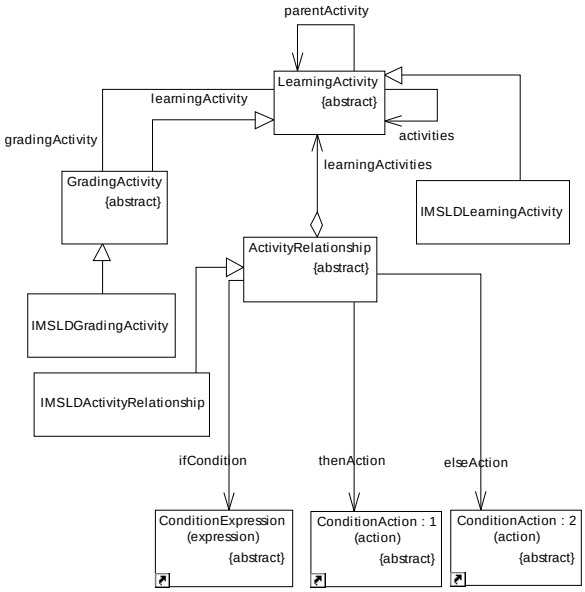

Figure 7. The model of learning activities in the instructional design description

3.5. IMS LD support

Described classes represent a general model of an e-learning course and consequently they not define any specific output format. In order to generate a course in the specific output format, it is necessary to create successors of the described classes. We have created successors that generate a course in the IMS LD format. For each learning element in the instructional design model, we have created an appropriate successor class (`IMSLDElementGroup`, `IMSLDSequence` and `IMSLDLearningObject`). Likewise, for conditions and actions, we have created successors that generate conditions and actions in the IMS LD format. All successors override `exportToXML` method and create the XML nodes in the IMS LD manifest file.

`LearningActivity` and `ActivityRelationship` are abstract classes and they represent activities in general. We have created successors – `IMSLDLearning-Activity` and `IMSLDActivityRelationship` in order to generate activities in the IMS LD format. These classes export their attributes to the IMS LD format.

4. SYSTEM OUTPUT

In this paper we have shown how to generate the course in the IMS LD format. But, the model enables generating a course in any XML-based format. It is only necessary to

implement appropriate successor classes that export the content to desired format.

So far the system is used for generating the e-course *Numerical Algorithms and Numerical Software in Engineering* at Faculty of Technical Sciences in Novi Sad. We applied three instructional strategies and generated three versions of our course. A part of the one of the generated IMS LD manifest files is shown in listing 1.

```
<imsld:learning-activity
       identifier="Iterativni_postupci_1">
   <imsld:title>
       Iterativni postupci
   </imsld:title>
   <imsld:activity-description>
      <imsld:item identifierref =
                  "iterativni_postupci_res">
        <imsld:title>
              Iterativni postupci
        </imsld:title>
      </imsld:item>
   </imsld:activity-description>
</imsld:learning-activity>
```

Listing 1. A part of the generated IMS LD manifest file

The created course may be shown in any IMS LD player and we used Reload LD Player [13]. A screenshot of the generated course is shown in Figure 8.

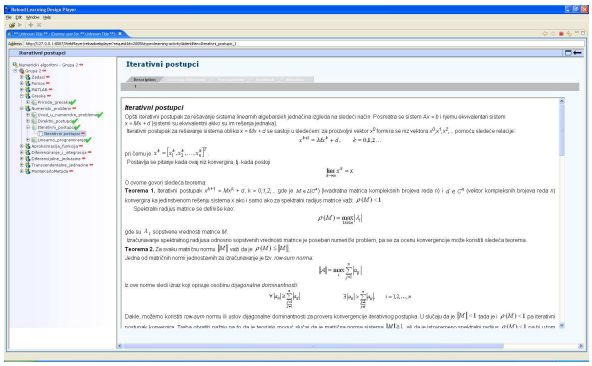

Figure 8. Screenshot of the generated course in the Reload LD Player

5. CONCLUSION

This paper describes the implementation of the system for automatic generation of e-learning courses. The system architecture, object model and functionalities are presented. The model contains two layers. First layer models an abstract course. The second one models a course in the concrete output format. Our system generates courses in the IMS LD format. The model contains following subcomponents: the model of learning objectives and learning objects, instructional design model and the model of learning activities. System output is the IMS LD manifest file. The implementation is done in Java programming language.

By changing the input parameters, our system may generate different versions of an e-learning course. Still, the generated course is static – it contains a sequence of predefined learning activities. This could be taken as a drawback of our system. The system could be improved if the generated course chooses a next learning activity in the real-time. A student would dynamically get a learning activity depending on various parameters (instructional design, student's knowledge state, personal preferences ...). However, in this phase we have chosen to generate only a static course because most of the popular e-learning systems and e-learning standards don't have support for dynamic courses. Another disadvantage still remaining in our system is that it doesn't consider a student model, so the generated course is not personalized.

The system has been used at Faculty of Technical Sciences in Novi Sad for generating the course *Numerical Algorithms and Numerical Software in Engineering* which is presented as an illustrative example in this paper. In addition, we have generated the course *Web programming*. Our plan is to apply this e-course in the summer semester 2011.

Future work is concerned with analysis of the data collected while the generated course is used in the teaching process. Our short-term goal is to find the most appropriate instructional strategy for the *Numerical Algorithms and Numerical Software in Engineering* course. Also, the plan is to create graphical tools for defining input parameters. So far, we have created a graphical editor for defining course instructional design and we are implementing a graphical editor of learning objectives management. Long-term goal is to develop a system supporting dynamic courses and containing a student model as the input parameter. Using student's knowledge state and personal preferences, the system would generate a personalized e-course.

6. REFERENCES

[1] Brusilovsky, P., Vassileva, J., *Course sequencing techniques for large-scale web-based education*, Int. J. Cont. Engineering Education and Lifelong Learning, Vol. 13, Nos. 1/2, 2003.

[2] Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., Ullrich, C., *ActiveMath: A Generic and Adaptive Web-Based Learning Environment*, International Journal of Artificial Intelligence in Education, (12), 2001.

[3] Kettel, L., Thomson, J., Greer, J., *Generating Individualized Hypermedia application*, Proc. Of Workshop on Adaptive and Intelligent Web-based Education Systems at 5th International Conference on Intelligent Tutoring Systems, Montreal, Canada, 2000.

[4] Kontopoulos, E., Vrakas, D., Kokkoras, F., Bassiliades, N., Vlahavas, I., *An ontology-based planning system for e-course generation*, Expert Syst. Appl. 35, no. 1-2, 2008.

[5] Hernandez, J., Baldiris, S., Santos, O.C., Fabregat, R., Boticario, J. G., *Conditional ims learning design generation using user modeling and planning techniques*, Proceedings of the 2009 Ninth IEEE

International Conference on Advanced Learning Technologies, IEEE Computer Society, 2009.
[6] Capuano, N., Gaeta, M., Micarelli, A., Sangineto, E., *An integrated architecture for automatic course generation*, Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 02), 2002.
[7] Lluvia, M., Luis, C., Juan, F.-O., Arturo, G.-F., *Automatic generation of user adapted learning designs: An AI-planning proposal*, Proceedings of the 5th international conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer-Verlag, Hannover, Germany, 2008.
[8] Segedinac, M., Savić, G., Konjović, Z., *Knowledge representation framework for curriculum development*, International conference on knowledge engineering and ontology development, Valensia, Spain, 2010.
[9] Tyler, R., *Basic Principles of Curriculum and Instruction*, Chicago: University of Chicago Press, 1949.
[10] W3C, *OWL Web Ontology Language Overview*, 2004.
[11] IMS, *IMS content packaging specification*, 2007.
[12] IMS, *IMS Learning Design Information model*, 2003.
[13] Reload, Learning design player v. 2.1.3, 2010, Internet site: www.reload.ac.uk/ldplayer.html

Acknowledgments

Results presented in this paper are part of the research conducted within the Grant No. III-43007, Ministry of Science and Technological Development of the Republic of Serbia.