Building Learning Designs by Using an Automatic Planning Domain Generation: A State-Based Approach

Lluvia MORALES [a,e], Luis CASTILLO [b,e] Juan FERNANDEZ-OLIVARES [c,e], Arturo GONZALEZ-FERRER [d,e]

[a] *lluviamorales@ugr.es*
[b] *L.Castillo@decsai.ugr.es*
[c] *faro@decsai.ugr.es*
[d] *arturogf@ugr.es*
[e] *Department of Computer Science, University of Granda, Spain*

Abstract. A Learning Design definition under the IMS-LD standard is a complex task for the instructor because it requires a lot of time, effort and previous knowledge of the students group over which will be defined the knowledge objectives. That is why, taking advantage from diffusion of learning objects labeling using IMS-MD standard in distance learning field, we have proposed to realize a knowledge engineering process, represented as an algorithm, with learning objects labels and user models to automaticaly define a domain that will be used by an intelligent planner to build a learning design. This learning design will be finally implemented in the ILIAS Learning Management System.

Keywords. Planning and Scheduling, e-learning, IMS standars, Automatic Generation of Planning Domains

Introduction

Since the appearance in 2003 of the IMS-LD standard final version endorsed by IMS Global Consortium[8], lot of educators have tried to implement it within on-line Learning Managment Systems(LMS) they use. However, this implementation is not an easy work because learning objects have to be completely labeled[1] and it is necessary to detail the process to use them in order to achieve each student objectives, that is, specify a learning design(LD).

Learning objects labeling, despite the initial cost it implies, is a widely accepted work in current educational community. On the other hand, most of educators have no training in learning design generation and does not have enough time and/or necessary students group knowledge to detail it since the beginning of the course. For this reason, researchers have actually being looking for techniques to facilitate and even skip the learning design construction step that commonly is assigned to the tutor.

We propose, on the one hand, to take advantage of the already labeled learning objects in a particular subject and, on the other hand, of the initial description of main stu-

dent characteristics (user model) to, through a knowledge engineering process traduced in an algorithm, automatically obtain a detailed description of the learning environment adapted to the students requirements. This learning environment is called domain and is written in the planning domain definition language (PDDL)[10] used by an intelligent planner to generate an adaptive learning process which is part of the particular student learning design. This finally must be used by a LMS, ILIAS in this specific case.

Through this paper we will first analyze the previous work made in realtion with the learning design automatic generation and the advantages of our approach. Secondly we describe the problem we face and why the necessity to automatically generate learning designs. Subsequently, we will explain the learning design construction process starting with the mandatory learning object metadata and student description, following with planning domain and problem characteristics description stipulated to LPG planner[5], and concluding with the knowledge extraction and domain generation algorithm which have been adapted to satisfy the LPG-td planner requirements in order to automatically construct the learning design. We also explain how to adapt this learning design to the ILIAS-LMS[7], and finally we mention the conclusions we have obtained through the experimentation and future work lines.

1. Related Work

Since 1986 Peachy and McCalla[13] proposed the integration of planning and scheduling techniques in order to improve the course path from Intelligent Tutoring Systems (ITS's). Last year two works in this field have appeared: first, Kontopoulos, et al.[9] created PASER which uses ontologies over learning objects curricula, after the planning process, to support the lack of information about its relations. However, Kontopoulos does not use temporal reasoning and tutor interaction which is used by Camacho, et al.[2] in order to reorganize the contents of an e-learning course after a test made by the student. The last approach works over e-learning courses and adapt the plan to the IMS-LD standard[8], but it does not take advantage of the standards at all.

Lot of work have been done through those 22 years of research in the field, and McCalla, et al.[11] have recently proposed to use an exhaustive IMS-MD[1] standard labeling over learning objects for the definition of a domain model. This paper also shares the stadards claim, but we propose to do it by using a knowledge engineering algorithm for directly generate the planning domain of a course in the Planning Domain Definition Language[10], so that we can easily use a state-of-the-art planner to solve the problem.

In this case we have used the LPG-td planner because of its support of numerical quantities and durations that allow us the generation and adaptation of plans. This characteristic is required to solve the problem of the learning design adaptation for every student.

2. The Adaptive LD Construction Problem

In IMS-LD standard three representation levels of a learning design are described which represent themselves a big problem:

Level A Implies definition of different roles that will be part of the learning environment design and of the actions that will be done inside the environment.

Level B Works over the definition of personalized learning units according to different pedagogies. This take into account the reusability of learning objects, the previous knowledge of each student and his preferences.

Level C Allows the communication between system components and roles, accept student profile changes that involve a learning unit redefinition and the sending of messages to other roles. This actions are done in real time and in a collaborative way.

Our objective is to make easier the job of exhaustively analyze the student characteristics and, after that, to define the better educational objective for each one. Those actions are described in level B as a tutor work, but we are deleting this job from his programme of work with our proposal.

Originally, as we have seen at the previous paragraph, is the tutor who must specify the learning path for each student, but this requires to invest lot of training, time, patience and analytical effort. A few years later appeared some proposals that applied intelligent planning techniques to resolve the work of defining the learning path; but despite liberating the tutors doing the work, the analysis and knowledge extraction work was transferred to a planning expert who had to deal with several subjects at the same time: to generate planning domains for each one, and to define planning problems according to each user model.

Actually, we are making real the idea of replacing with an algorithm the knowledge engineering process that would be made by a planning and scheduling expert. The algorithm extracts the knowledge taking advantage of the reusability of learning objects labeling, based on IMS-MD standard, and generates a planning domain with this knowledge. The metadata labeling will be addressed in the next section.

3. LD Automatic Construction Using AI Planning

In order to be able to start with the Learning Design generation process is essential to extract mandatory information about learning objects of the subject using its metadata and the user models of each one of the students registered. Then an algorithm may use that information in the planning domain definition and finally the LPG-td planner will get an adapted learning design for every student.

3.1. IMS-MD

Learning objects have a lot of metadata that may be completely described to satisfy the IMS-MD standard. However, it is enough to get the following metadata that we have divided in two types according to its characteristics, for the knowledge engineering process:

Hierarchy relations metadatas:

1. *Is-Part-Of*. It describes a hierarchical compositional structure between learning objects through the course presented in figure 1 where, for example *Algorithms* `Is-Part-Of` *Discrete Maths*.

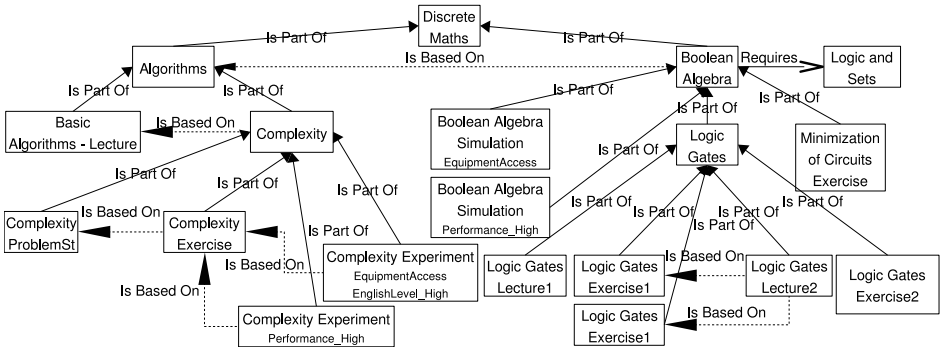

Figure 1. Hierarchy Relations in a Subject

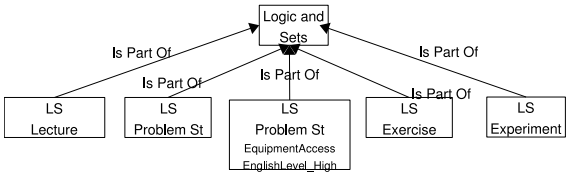

Figure 2. Required Object

2. *Is-Based-On*, provides ordering relations between primitive objects or compound objects like *Logic Gates Lecture2* `Is-Based-On` *Logic Gates Problem St*, or *Complexity* `Is-Based-On` *Basic Algorithms Lecture*.
3. *Requires*. Reports content dependencies. Usually when a compound object like *Boolean Algebra* in figure 1 needs another one from an extern course like *Logic and Sets* totally described in figure 2.

Objects Attributes Metadatas:

1. *Language*. Defines the langugae of the learning object. It could be spanish, english, french, etc.
2. *Learning Resource Type*. Is an educational metadata which describes what kind of learning resource we are working on, for example: lecture, problem statement, simulation, experiment or exercise.
3. *Other Platform Requirements*. Describes if there are special hardware or software requirements to use the learning object.
4. *Difficulty*. Metadata that defines the performance level that is required by the student for this object to be in his plan.

As we can see in figures 1 and 2 a subject has learning objects(that we call primitive objects) which are part of another learning objects(compound objects). This primitive objects have to be realized in a particular way given by `Is-Based-On` relations or its `learning resource type` in order to satisfy each compound object learning requirements and there are relations between compound objects too that helps us to finally complete the subject learning path.

3.2. User General Model

Usually portfolios or personal data sections in learning managment systems have a wide range of options that can be used by the student to describe his/her profile. But, in order to personalize the learning design, our algorithm use the next ones:

1. *English Level*. To determine if he could take a high level english object.
2. *Equipment*. Defines software and hardware disposition of the student like flash plugins, java environment, type of screen, bandwidth, etc.
3. *Previous Courses Level*. Score in a related course. According to this metadatum a required task could be taken or not.
4. *Performance Level*. Performance level of the student. If high, he could do difficult learning objects.
5. *Learning Style*. It help us to offer each user a set of learning objects with a temporal sequence that best fits his/her learning style defined by a psicological test that is answered by the student in his/her first visit to the LMS. At this time we are using Honey-Alonso test[6] but a Felder test or another kind of learning style definition test might be used.

3.3. The Automatic Construction Process

In this section we show a brief description of the basic elements to the intelligent planner domain and problem construction; a domain specifically designed for the LPG-td state-based planner. Subsequently the algorithm to obtain and organize those elements to finally form PDDL documents(domain and problem) is explained.

3.3.1. LPG State-Based Planner

An intelligent planner needs to represent the objects in a certain way (hierarchical or state-based) to establish order relations between them using one o several methods.

LPG-td is a state-based intelligent planner based on local search and planning graphs methods that handles PDDL2.2 domains involving numerical quantities and durations[4]. The system can solve both plan generation and plan adaptation problems like LD generation, and have to consider the following assumptions to define its problem and domain definition:

* The initial state of the planning problem is based on the contextual information extracted from LMS databases like user profiles and academic history.
* The goal of the planning problem is translated from the learning objectives of a given course, in this case the last learning object(s) needed to complete the subject.
* The set of the available actions in the domain is built from the learning objects repository so that every primitive object is translated into an action whose preconditions and effects are inherited from the information expressed in its metadata.
  For example, as we can see in figure 1, one of the preconditions from *Complexity Experiment* primitive object is that *Complexity Exercise* has to be finished previously, another precondition can be a high performance, and, if preconditions are satisfied, the effect will be that *Complexity Simulation* object can be done and marked as finished as in figure 3.

```
( :action actionComplexity-Experiment
    :parameters (?s - Student)
    :precondition
      (and
        (performance_High ?who)
        (actionComplexity-Exercise_done ?who) )
    :effect
      (actionComplexity-Experiment_done ?who)
 )
```

Figure 3. Action that shows "simple" precondtions for an action without `Learning Style` preconditions

.

The last point is the base of our next domain generation algorithm which makes a special emphasis in order preconditions. Order preconditions help us to define the object(s) which are preconditions of other objects and are needed to produce the desired effect, while the action effect(the actions who needs preconditions) is made once preconditions are true.

3.3.2. The Domain Generation Algorithm

The planning domain generation algorithm is responsible of specifying required preconditions for every learning object. These learning objects are defined and labeled by the instructor in order to be used by the student in the better way according to the learning objects restrictions and his/her learning style.

The algorithm first analyze attributes and relations from each object, secondly it defines preconditions for primitive objects according to the requirements given by hierarchy relations, and finally it defines preconditions about order relations between objects.

This last step is really arduous because it implies to check a subject graph from primitive objects, forming groups linearly arranged (even in a parallel way because of objects with the same name). These groups (called primitive groups) are ordered according to the `Is-Based-On` relation or the learning style, and subsequently we rise to each one of the hierarchical levels occupied by compound objects and inheriting its relations to the first primitive object of the primitive group previously formed that is part of this compound. With this inheritance relations we can reorder those groups and form new ones.

The process described in the previous paragraph is done till cover every order relation from compound objects which are part of a root compound object(*Discrete Maths* in figure 1 o *Logic and Sets* in 2) and finally we have to connect this root objects according to `Requires` relations.

The domain generation algorithm is the next one:

I Analyze
  * Primitive objects attributes. Then depending on the requirements, expressed on the attributes, may be possible to extract the preconditions for each one.
  * Objects Relations. To continue with the next step of the algorithm.

II Order

  1 Primitive Objects
    For each one of the primitive objects groups part of the same compound object, make the next corresponding process in order to obtain the order preconditions to each primitive object of the group:

```
( :action actionLS-Excercise
    :parameters (?who - Student)
    :precondition
      (or
        (and
          (LSTheoretical ?who) (actionLSExperiment_done ?who) )
        (and
          (LSPragmatical ?who) (actionLS-Lecture_done ?who) ) )
    :effect
      (actionLS-Exercise_done ?who)
)
```

Figure 4. Action that shows "simple" and `Learning Style` preconditions

.

1.1 If objects are linear and completely sorted according to the `Is-Based-On` relation (as children from *Complexity* in figure 1), the group may be ordered following the indicated by `Is-Based-On` relation order. Then the precondition from each object will be that its previous object(s) in the new ordered group has to finished. Example: if *A* `Is-Based-On` *B*, *B finished* is the precondition of *A* because contains key information to understand *A*.

1.2 If objects have not `Is-Based-On` relations between them, as in figure *Logic and Sets* children from figure 2, order the objects using its `learning resource type`(LRT) and the next rules given by the Honey-Alonso`Learning Style`:

* If learning style is `theoretical` primitive objects with problem statement `LRT` will be in the first position of the new ordered group, objects with simulation `LRT` secondly, experiment `LRT` objects after them, exercise `LRT` and finally objects with lecture `LRT`.

* If the learning style is pragmatical simulation, experiment, problem statement, exercise and lecture `LRT` is the order in which the objects will be ordered.

The precondition for each object will be *"The previous object in theoretical group has to be finished and student must have theoretical learning style, or the previous object in pragmatical group has to be finished and student must have pragmatical learning style"*.

The PDDL code in figure 4 is an example of how an action can have different actions as preconditions depending of the Learning Style of the student we are making the plan.

1.3 If primitive objects of the group have some `Is-Based-On` relations between them but the order is not total:

* Find inside this group linear totally ordered by `Is-Based-On` subgroups, like Logic Gates Lecture2 and Logic Gates ProblemSt in figure 1, and order each one of this subgroups as in 1.1.

* Find no ordered objects (objects that have no `Is-Based-On` relation or are not the base of anyone) in the principal group, like Logic Gates Exercise and Logic Gates Lecture in figure 1, and order each one of them as in 1.2.

2 Groups and/or Primitive Objects in Higher Levels

To order this objects we inherited relation attributes from the group of primitives' parent to the first object of the group, using the `Is-Part-Of` relation.

For example: `Is-Based-On` relation that relates *Complexity* with *Basic Algorithms* is inherited to *Complexity ProblemSt* in figure 1; this relation will be used to reorder primitive objects in this level using step 1 rules and the next two tips to assign preconditions according to its variations.
- If an ordered group of primitive objects previously ordered is before a lonely primitive object, the last object of the back group will be the precondition of the lonely object, as Complexity Simulation finished could be precondition of *Boolean Algebra Lecture* for example.
- If a group of ordered primitive objects *A* is before to other group *B*, then the last primitive object finished of *A* will be the precondition of the first object of *B*.
This reordering of groups will be carried out until every order relation from compound objects part of a root compound objects (as *Discrete Maths*) be inherited to primitive objects of primitives groups, and preconditions between groups have been defined.

3 Required Objects
When a compound object *A* requires an extern compound object *B*, the last primitive object of *B finished* will be the precondition the first primitive object of compound *A*.

Using this process is possible to generate a planning domain in PDDL with the same expressive capacity than the IMS-MD repository.

3.3.3. Problem Generation

The problem file is extracted in an automated manner from each student characteristics mentioned in section 3.2, which are translated to predicates that the domain will be able to evaluate. When a problem achieve its goal is because it satisfied the last activities of the course, that is, objects that are not required by any other to satisfy the basic needs of the course.

In figure 5 three different student models are described including predicates used by the PDDL problem file.

3.3.4. LD Generation by LPG

Once domain and problem PDDL files are generated, the LPG planner creates the user profile adaptive learning designs to the subject we are making the domain.

In figure 6 there are the examples of automatically generated Learning Designs to the student models described in figure 5. Those learning designs are adapted to each student: Jhon has a `theoretical` ordered plan, but he needs easier learning objects in his learning design because of his low `performance`; Chris has a `theoretical` LD too, his learning objects can be difficult resources, but can not take objects with `equipment` needs and he `requires` to take *Logic and Sets* primitive objects; finally, a `pragmatical` LD was automatically generated to Thom, he only needs to take *Logic and Sets* because he does not have any other limitation.

3.4. LD Integration in ILIAS

The procedure described along this paper has been implemented in Python and fully integrated in the ILIAS LMS, which embeds a SOAP (Simple Object Access Protocol)

Jhon	
LSTheoretical Performance_Low EquipmentAccess EnglishLevel_High PreviuosLogicAndSets	Jhon is a deductive student because his theoretical learning resource type. He has a low performance, but whit a high english level, the Logic and Sets subject finished, and whit the necessary equipment available.
Chris	
LSTheoretical Performance_High EquipmentAccess_No EnglishLevel_High PreviousLogicAndSets_No	Chris is a deductive student whit a high performance and english level, but he has not finished the Logic and Sets subject and he has not the necessary equipment available.
Thom	
LSPragmatical Performance_High EquipmentAccess EnglishLevel_High PrevoiusLogicAndSets_No	Thom is an inductive student because his pragmatical learning resource type. He has high performance and english level. He can access to the necessary equipment, but he has not finished the Logic and Sets subject.

Figure 5. Three Studied Students - Problem Definition

Chris		
	Learning – Primitive Object	*Requirements*
1	Basic Algoritms - Lecture	
2	Complexity - ProblemSt	
3	Complexity - Exercise	
4	Complexity - Experiment	Performance
5	LS - Lecture	
6	LS - Experiment	
7	LS - Exercise	
8	LS - ProblemSt	
9	Boolean Algebra - Simulation	Performance
10	Minimization Of Circuits - Exercise	
11	Logic Gates - Exercise2	
12	Logic Gates - Lecture1	
13	Logic Gates Exercise1	
14	Logic Gates - Lecture2	

Thom		
	Learning – Primitive Object	*Requirements*
1	Basic Algoritms - Lecture	
2	Complexity - ProblemSt	
3	Complexity - Exercise	
4	Complexity - Experiment	Equipment, english
5	LS - Experiment	
6	LS - Lecture	
7	LS - Exercise	
8	LS - ProblemSt	
9	Boolean Algebra - Simulation	Equipment
10	Minimization Of Circuits - Exercise	
11	Logic Gates - Exercise2	
12	Logic Gates - Lecture1	
13	Logic Gates Exercise1	
14	Logic Gates - Lecture2	

Jhon		
	Learning – Primitive Object	*Requirements*
1	Basic Algoritms - Lecture	
2	Complexity - ProblemSt	
3	Complexity - Exercise	
4	Complexity - Experiment	Equipment, english
5	Boolean Algebra - Simulation	Equipment
6	Minimization Of Circuits - Exercise	
7	Logic Gates - Exercise2	
8	Logic Gates - Lecture1	
9	Logic Gates Exercise1	
10	Logic Gates - Lecture2	

Figure 6. Learning Designs from Students in figure 5

server, so that several Python scripts implement the extraction procedures described so far, just by using the available SOAP functions, and obtain the domain and problem files. The LPG-td planner is then executed and a plan is obtained. ILIAS does not support IMS-LD specification yet, so in order to make the plan available to student, we have translated the plan into a follow up guideline that appears over the student' ILIAS desktop.

4. Concluding Remarks and Future Work

As we can observe throughout this paper, we get to make the most of the standard IMS-MD labeling to, through a knowledge extraction process, define a planning domain that is used by a planner for the later generation of a plan adapted to the need of each user of a specific course, which can also be called LD (in its abstraction level B).

It is important to point out three of this paper. First, that the labeling extraction can

be carried out independently of the course over which our LD is going to be created. Second, that an intelligent state-based planner has been used. Finally, that the knowledge extraction process is carried out by an algorithm (designed by our research team) that does not require the participation of any planning expert.

The final product after this process, the LD, must be supervised by the instructor to make the considered modifications, in case it is needed.

Throughout the realization of this research work, we realized that it is possible to improve it. The next improvements can be achieved in short and mid-term.

1. To design a domain representation that works out the common characteristics of a user group to be able to generate a collaborative LD. In this aspect, it would be needed to obtain and give details of some aspects of the planning problem which are not trivially achieved from the user models.
2. To apply continual planning [3][12] within the adaptive LD planning, pursuing that the system can use input of run-time information, and include within the domain different roles and/or messages that should be sent when the students complete a specific activity.
3. To identify the way to work with different kinds of Learning Objects (optionals, for example) and to represent global and partial deadlines that can be managed by a state-based planner.

References

[1] ANSI-IEEE. *IEEE Standard for Learning Objects Metadata* http://ltsc.ieee.org/wg12/.

[2] D. Camacho, M.D. R-Moreno, and U. Obieta *CAMOU: A simple integrated eLearning and planning techniques tool.* In: Proceedings of the CONTEXT'07 Workshop on Representation Models and Techniques for Improving E-Learning, August 2007.

[3] M. Fox, A. Gerevini, D. Long, and I. Serina, *Plan Stability: Replanning versus Plan Repair*, In: Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS'06), 2006

[4] A. Gerevini and I. Serina, *LPG: a Planner based on Local Search for Planning Graphs*. In: Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02), AAAI Press, Toulouse, France, 2002.

[5] A. Gerevini, A. Saetti, I. Sirina, and P. Tonielli *Planning in PDDL 2.2 domains with LPG-TD.* In: International Planning Competition booklet (ICAPS-4), 2004.

[6] *Honey Alonso Learning Style* http://www.estilosdeaprendizaje.es/chaea/chaea.htm

[7] *ILIAS Learning Management System.* http://www.ilias.de/ios/index-e.html

[8] *IMS-GLC IMS Global consortium.* http://www.imsglobal.org/.

[9] E. Kontopoulos, D. Vrakas, etal. *An Ontology-based Planning System for e-course Generation*. In: Expert Systems with Applications, Elsevier. Available online July 2007.

[10] D. Long and M. Fox *PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains.* In: Journal of Artificial Intelligence Research, Volume 20:61-124, 2003.

[11] P. Mohan, J. Greer, and G. McCalla. *Instructional planning with learning objects.* In: IJCAI-03 Workshop Knowledge Representation and Automated Reasoning for E-Learning Systems, 2003.

[12] K.L. Myers *Towards a Framework for Continuous Planning and Execution.* In: Proceedings of the AAAI Fall Symposium on Distributed Continual Planning, 1998.

[13] D.R. Peachey and G.I. McCalla *Using Planning Techniques in Intelligent Tutoring Systems*, International Journal of Man-Machine Studies Volume 24:77-98, 1986.