The Knowledge Engineering Review, Vol. 28:2, 121–136. © Cambridge University Press, 2013
doi:10.1017/S0269888912000380
First published online 8 February 2013

On the automatic compilation of e-learning models to planning

ANTONIO GARRIDO[1], SUSANA FERNÁNDEZ[2], LLUVIA MORALES[3], EVA ONAINDÍA[1], DANIEL BORRAJO[2] and LUIS CASTILLO[3]

[1]*Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera s/n, 46071 Valencia, Spain;*
e-mail: agarridot@dsic.upv.es, onaindia@dsic.upv.es;
[2]*Departamento de Informática, Universidad Carlos III de Madrid, Avenida de la Universidad 30, 28911 Leganés, Madrid, Spain;*
e-mail: sfarregu@inf.uc3m.es, dborrajo@inf.uc3m.es;
[3]*Departamento de Ciencias de la Computación e I.A., Universidad de Granada, C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain;*
e-mail: lluviamorales@decsai.ugr.es, l.castillo@decsai.ugr.es

Abstract

This paper presents a general approach to automatically compile e-learning models to planning, allowing us to easily generate plans, in the form of learning designs, by using existing domain-independent planners. The idea is to compile, first, a course defined in a standard e-learning language into a planning domain, and, second, a file containing students learning information into a planning problem. We provide a common compilation and extend it to three particular approaches that cover a full spectrum of planning paradigms, which increases the possibilities of using current planners: (i) hierarchical, (ii) including PDDL (Planning Domain Definition Language) actions with conditional effects and (iii) including PDDL durative actions. The learning designs are automatically generated from the plans and can be uploaded, and subsequently executed, by learning management platforms. We also provide an extensive analysis of the e-learning metadata specification required for planning, and the pros and cons on the knowledge engineering procedures used in each of the three compilations. Finally, we include some qualitative and quantitative experimentation of the compilations in several domain-independent planners to measure its scalability and applicability.

1 Introduction

E-learning is becoming a high-impact innovative topic as it offers a promising way to facilitate and enhance the learning process by combining learning objects (LOs), typically any sort of digital resource, to create flexible courses. This flexibility and the availability of automatic tools to create learning routes allows e-learning experts to focus on content authoring, and relieves them of the burden of manually composing standard documents, such as learning designs, that fit the needs of each student in a course.

Creating tailored routes of LOs, according to student profile preferences and/or pedagogical theories, is a subject which has been studied in depth in the Planning & Scheduling (P&S) community within the last years (Mohan *et al.*, 2003; Kontopoulos *et al.*, 2008; Ullrich & Melis, 2009; Castillo *et al.*, 2010). Generating these routes depends on many elements: LOs prerequisites/outcomes, LOs duration, available resources, and even collaboration and interaction between tutors and students, which make the problem very interesting from the P&S perspective. However, acquiring

information from educational domains to represent it as a planning domain is not a straightforward task because pedagogical theories are usually hard to model in practice. The main reasons for this are: (i) there exist essential elements (e.g. soft and hard requirements among LOs, deadlines, strong interaction between LOs or students, etc.) for the success of the learning process, but are not thoroughly modelled in traditional e-learning settings; (ii) people give different meanings and uses to the standard specifications, mainly in terms of the relations among LOs, given that standards provide some flexibility on how to represent knowledge; and (iii) there is a great variety of planning paradigms that support different aspects of expressiveness. In this paper we explicitly address these drawbacks by providing an exhaustive mapping to interpret LO metadata in terms of artificial intelligence (AI) planning, which can be easily extended to support more complex features, not usually modelled in e-learning, and can be applied in different planning paradigms.

Leaving the complex pedagogical decisions aside, in this paper we propose an alternative to previous work that focuses on solving these three difficulties. We extend the work in Garrido *et al.*, (2009), presenting a knowledge engineering approach to compile information about course content and learners, subject to e-learning standards (IMSLD, 2003; IMSMD, 2003; Sharable Content Object Reference Model (SCORM), 2004), and translating this information into a planning domain and problem. The planning instances are used to automatically generate plans, customized routes of LOs, that are translated into an IMS-LD (IMS learning design) standard document (IMSLD, 2003). In essence, this paper contributes with:

- An automated translation of e-learning templates into a general compilation, further extended with three different PDDL (Planning Domain Definition Language (Fox & Long, 2003)) versions: (i) hierarchical, (ii) PDDL-conditional, and (iii) PDDL-temporal. This allows us to use our compilations in virtually any current planner.
- An effective use of planning technology to generate learning designs that best suit students learning goals. Planners focus on finding the adequate LOs combination for students adaptation, thus promoting a more personalized access to the LOs.
- A translator that parses the resulting plans and generates the IMS-LD standard document to be uploaded to Learning Management Systems (LMSs), thus closing the e-learning cycle.

This paper is structured as follows. Section 2 introduces some related work on courseware generation using planning techniques, some basic description on e-learning standards and motivates our work. Section 3 analyses how to model learning designs in planning, and presents the compilation of e-learning standards into planning domains and problems. Experimental results, by using different planners, are detailed in Section 4. Finally, Section 5 concludes the paper and addresses some future work.

2 Combining e-learning and artificial intelligence planning: motivation

There are many approaches in the adaptive hypermedia community that combine instructional knowledge and planning techniques to deal with the automation of courseware generation. Some of them introduce Hierarchical Task Network (HTN) planners to represent pedagogical objectives to find a tailored course structure, such as Méndez *et al.* (2005) and Sicilia *et al.* (2006), which provides a theoretical proposal. To our knowledge, the former does not integrate its results in an e-learning platform, whereas the latter only provides Learning Object Metadata (LOM) translations for HTN planning. The approach in Ullrich and Melis (2009) is complete and has been put into practice specially for retrieving LOs from external repositories. Basically, it takes a LOM subset to be integrated into a particular ontology with additional LO information by means of a specific intelligent tutoring system, thus making it incompatible with other learning systems. According to Ullrich and Melis (2009), they can map the final plan into IMS Content Packaging or SCORM but not the IMS-LD model itself and they do not use standards in the definition of the student profile. Others use state-based planners that implement metrics to measure the

adaptation to a specific learning style (Boticario & Santos, 2007; Limongelli *et al*., 2008), but without taking into consideration other profile features and different learning styles theories. Finally, others incorporate machine learning techniques to assist content providers when constructing LOs that comply with an ontology, concerning objectives and prerequisites, as can be seen in Camacho *et al*. (2007) and Kontopoulos *et al*. (2008) with CAMOU and PASER, respectively, but again, they do not use e-learning standards and usually focus on a particular planner. Despite the relative success of these approaches, they are usually limited to a specifically designed ontology and planning paradigm. Our main motivation (and contribution) is to overcome such limitations.

First, our approach is entirely based on well-known e-learning standards, and not on particular or *ad-hoc* ontologies. We use the IMS standard specification (`http://www.imsglobal.org`) supported by major LMSs such as Moodle (`http://moodle.org`) and dotLRN (`http://dotlrn.org`), which includes:

- IMS Meta-Data (IMS-MD) to describe LOs and their relations (see e.g. in Figure 2);
- IMS Learner Information Package (IMS-LIP) to model the student profile (see e.g. in Table 2); and
- IMS Learning Design (IMS-LD) to sequence the LOs according to the students profiles (see Section 3.4).

Second, we use a pedagogical theory based upon learning styles, but other adaptation criteria (see Essalmi *et al*., 2010) can be easily modelled as well. Hence, our approach provides the basis for education experts to easily experiment with different learning theories. The overall idea is to make the contents authoring easier and more adequate. As shown in Figure 1, the teachers design, from scratch or by reusing LO collections, an e-learning course. The course designer may use our authoring tool (see Garrido *et al*., 2009) to enrich the LO metadata by visually adding/deleting relations, or even associating resources and costs to the LOs. This is not thoroughly modelled in

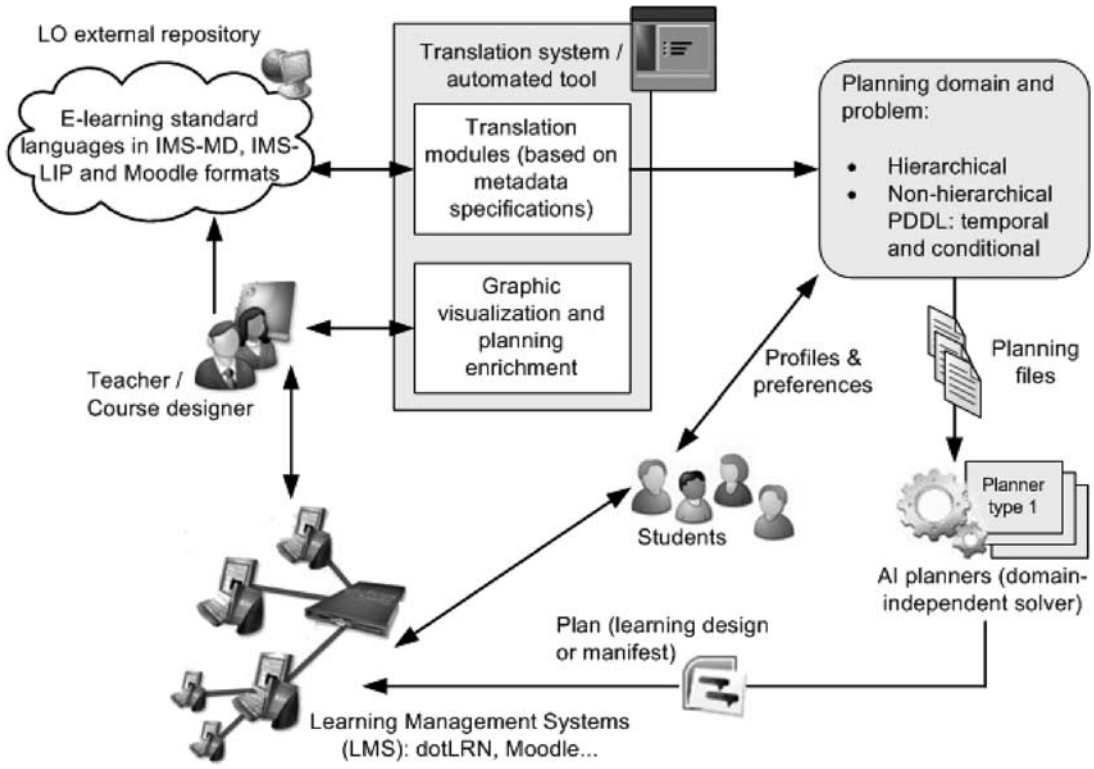

Figure 1 Overview of our entire system. LO = learning object; PDDL = Planning Domain Definition Language; LMS = Learning Management System.

```
<general>
  <identifier>Discrete-Maths</identifier>
  <title>
   <langstring xml:lang="en">Discrete-Maths</langstring>
  </title>
  <language>es</language></general>
<educational>
  <difficulty>
   <value>
    <langstring xml:lang="en">easy</langstring>
   </value></difficulty>
  <typicallearningtime>
   <datetime>0</datetime>
  </typicallearningtime></educational>
<relation>
  <kind>
   <value>
    <langstring xml:lang="en">Requires</langstring>
   </value></kind>
  <resource>
   <catalogentry>
    <langstring xml:lang="en">Algorithms</langstring>
   </catalogentry></resource></relation>
<relation>
  <kind>
   <value>
    <langstring xml:lang="en">Requires</langstring>
   </value></kind>
  <resource>
   <catalogentry>
    <langstring xml:lang="en">Boolean-Algebra</langstring>
   </catalogentry></resource></relation>
```

```
<general>
  <identifier>Basic-Algorithms</identifier>
  <title>
   <langstring xml:lang="en">Basic-Algorithms</langstring>
  </title>
  <language>es</language>
  <coverage>
   <langstring xml:lang="en">Mandatory</langstring>
  </coverage></general>
<educational>
  <learningresourcetype>
   <value>
    <langstring xml:lang="x-none">Narrative Text</langstring>
   </value> </learningresourcetype>
  <difficulty>
   <value>
    <langstring xml:lang="x-none">very easy</langstring>
   </value> </difficulty>
  <typicallearningtime>
   <datetime>9</datetime>
  </typicallearningtime></educational>
 <relation>
   <kind><value>
    <langstring xml:lang="x-none">IsPartOf</langstring>
   </value></kind>
   <resource>
    <catalogentry>
     <langstring xml:lang="en">Algorithms</langstring>
    </catalogentry></resource></relation>
```

Figure 2 Two learning objects (LOs) of an XML course. Irrelevant information has been ignored

traditional e-learning settings, but it makes the use of planning techniques more flexible; obviously, the enrichment of the metadata in our tool is entirely optional. The information about the course is profile-independent, so it is valid for students with different learning styles. Once the personal characteristics, background and preferences of the students are modelled, our translation system automatically generates the planning domain and problem files according to three different planning paradigms.

Third, our approach is appropriate for most of the existing planners, acting as domain-independent solvers. This is a clear advantage as we are not restricted to one particular solver. Once the solver generates a plan, as a LO route per student, it is subsequently compiled as an IMS-LD. Finally, the result is uploaded to a LMS, which manages the administration, display, tracking and navigation of the contents of the learning design, thus closing the cycle with standards.

3 Compiling e-learning models for planning

In our compilation we consider the repository of LOs as the planning domain, whereas the relevant students characteristics and interests represent the planning problem. On the other hand, the learning design represents the solution plan. In all cases, domain, problem and plan, the information required for the compilation is fully automated by extracting the metadata specifications from the e-learning standards.

3.1 Metadata for planning domains

A learning design is defined by a set of LOs, which are usually encoded as XML schemata (see Figure 2). For example, in a course for learning `Discrete-Maths`, there may be a task for learning `Boolean-Algebra`. And there could be several LOs to accomplish it, such as playing a visual presentation, reading the introduction text from a textbook or solving an exercise. Although it may be enough for the student to use only one of these objects to accomplish the task, it is still possible to use more than one, thus improving the overall utility (reward) of the learning process. And this usually depends on how the LOs are described by their metadata set:

- LOs are more or less appropriate to students depending on their learning styles. We use two well-known theories to determine and classify students, the Felder learning style[1] (Felder, 1996)

[1] Felder learning styles comprise four dimensions: perception (sensitive/intuitive), processing (active/reflective), input (visual/verbal) and understanding (sequential/global).

and the Honey-Alonso learning style[2] (Alonso & Honey, 2002), but any other theory can be used to describe new ordering rules or utilities. Thus, we use the representation of the pedagogical knowledge as defined in the metadata.

- LOs have dependency relations among them. For instance, before learning about `Discrete-Maths`, the student should have some knowledge on `Boolean-Algebra`. We support four types of relations that include hierarchical structures and content ordering relations. The hierarchical structures use the *IsPartOf* relation, which represents a hierarchical aggregation of LOs. Additionally, there are three types of causal dependencies, *Requires*, *IsBasedOn* and *References*. We interpret the first two relations as hard preconditions. In the case of the *Required* elements, all of them (conjunctively) have to be completed before initiating a new LO: if 'A *Requires* B' and 'A *Requires* C', both B and C need to be finished before doing A. In the case of the *IsBasedOn* elements, at least one of them (disjunctively) has to be completed: if 'A *IsBasedOn* B' and 'A *IsBasedOn* C', only B or C must be completed before initiating A. On the other hand, the course designer might also recommend (soft requirement) other previous LOs by using the *References* relation.
- Each LO takes a standard time (duration) to fulfill, namely typical learning time. However, only *primitive* LOs have duration, as the duration of aggregations is derived from their set of aggregated LOs (e.g. `Discrete-Maths` in Figure 2).
- Each LO belongs to a learning resource type, such as a lecture, narrative text, diagram, etc. This does not seem to be very relevant for planning, but according to education experts the learning resource type highly interacts, positive or negatively, with the student profile. For instance, a lecture is very recommendable for Felder verbal students but not for visual ones; and just the opposite holds for a diagram. Or LOs can be displayed in different orders; for example, for a Honey-Alonso pragmatic student, an experiment must be displayed before a narrative text, but for a theoretical student this order must be inverted. The main inconvenience here is that these resource types combine constructivist and traditional didactic approaches (Ullrich, 2008). That is, some describe the format of a resource, but others cover the instructional type, which represents different dimensions—there is a mix of pedagogical and technical/presentation information. Therefore, the learning resource type may fail in representing a sufficiently precise compilation if, for instance, a lecture contains an image or diagram, or an experiment is described as a text. This may become problematic, but we rely our technical compilation on the resource type, and delegate its significance entirely to the LO designer, who is the real expert in pedagogic matters.

3.2 Planning domain compilation

The general algorithm to compile a planning domain is to iterate all over the LOs and generate one action (or operator) per LO. This generation relies on a closed world assumption, and if new LOs are to be used the domain must be recompiled. But note that only the LOs that are affected by the changes need to be recompiled. For instance, assuming a domain of 10 LOs, which is extended with two new LOs that only affect (i.e. are related to) one of the original LOs, we only need to recompile and add $2 + 1 = 3$ LOs. That is, although the LO repositories may change frequently, in most cases the already generated domains may remain valid and we only need to add the new LOs. This compilation is very quick, as each action only consists of five entries—name, parameters, duration, preconditions and effects—which are automatically extracted from the values of the LO metadata specification according to the mapping of Table 1:

- The name needs to be unique, which is not particularly hard to generate, because two actions cannot have the same name.

[2] This learning styles theory considers four styles that can be parallelized; for instance pragmatic/active and reflexive/theoretical can be taken as two styles.

Table 1 General LO metadata mapping to PDDL actions—irrelevant information has been ignored

LO metadata item →	PDDL action entry
general/{`identifier` or `title`}	`action-name`, which may be a real, fictitious, primitive or an aggregation action (also known as task in HTN)
-	:`parameters` (`?s - student`), to model the student

if the LO is a *primitive* LO then
`educational/typicallearningtime`
else {*aggregation of LOs*}
sum of the `educational/typicallearningtime`
given by its *IsPartOf* LOs
	:`duration` for temporal planners, or
	:`effect` (`increase` (`total_time ?s`) *value*)
	managed as an artificial fluent effect for temporal and non-temporal metric planners

dependency relations: type of relation: switch (`relation/kind/value`) case (*Requires*): conjunctive (and) precondition case (*IsBasedOn*): disjunctive (or) precondition the related LO: `relation/resource/catalogentry`	:`precondition` (`and` (`not` (`action-name_done ?s`)), and if conjunctive precondition: (`and` … else-if disjunctive precondition: (`or` … for each `catalogentry` *value*: (`action-`*value*`_done ?s`), which creates a particular ordering (also required in HTN) … other optional preconditions for profile adaptation taken from `educational` item)
increase a reward (utility expression) or creating a particular ordering (useful in HTN) due to: profile adaptation: `educational/learningresourcetype/value` additional adaptation by using a metric reward: if `relation/kind/value` is *References*	:`effect` (`and` (`action-name_done ?s`) …, to model that the action has been done, and (`increase` (`reward ?s`) *value_LRT*) (`increase` (`reward ?s`) *value_References*) … other optional rewards and/or costs)

LO = learning object; PDDL = Planning Domain Definition Language; HTN = Hierarchical Task Network.

- The use of a parameterized student facilitates the application of this action to different students, and makes the definition of preconditions/effects more flexible.
- The action duration is calculated to model either a durative action, for example, as defined in level 3 of PDDL2.1 (Fox & Long, 2003), or as a numeric-valued fluent for metric planning optimization.
- The preconditions support all the dependency relations, according to the semantics of conjunctive (*Requires*) and disjunctive (*IsBasedOn*) preconditions. A precondition (not (`action-name_done ?s`)) is used to avoid planning the same action more than once. Further, other educational requirements, such as the intended role of the student or the difficulty of the LO, can be easily modelled.
- The effects encode the fact of attaining the outcome of the LO, that is, having the action done. They also include a `reward` to offer a full support for LO adaptation to the students. This is achieved in two ways. First, modelling both the learning resource type of each LO and the student learning style, based on Felder classification, Honey-Alonso or any other classification. Using recommendation tables, such as Baldiris *et al.* (2008), we know how good (*value_LRT*) a given LO is for a learning style. Second, the *References* dependency relation also may increase the reward by a given *value_LRT* when the student satisfies that recommendation. Again, this value can be particularized to each learning style. Optionally, the compilation can include numeric expressions or resource costs, as are common in P&S, and even a particular sequencing used to assist hierarchical decomposition.

This mapping is very general in terms of the planning concepts. Although it comprises many features, such as durations, conjunctive and disjunctive preconditions, propositional and

numeric-valued representations, etc., it does not define other features such as hierarchical structures. On the other hand, there are currently several planning paradigms and planners. Thus, we instantiate this general algorithm to provide specialized compilations for different planning paradigms. More particularly, we started with an extension of our general compilation to a hierarchical compilation, and then we added two approaches for PDDL compilations. The first one based on conditional effects, named conditional compilation, that encodes in one operator all the profile adaptation options. Again, as many planners do not support conditional effects, we provide an additional compilation where all operators are grounded, hierarchies are simulated by means of dummy actions and actions can be durative (temporal compilation). It is important to note that we are not proposing three different planning systems, but three compilations aiming at covering practically the full spectrum of planning paradigms that exist today.

3.2.1 Hierarchical domain compilation

This compilation extends the general compilation with the following features:

- It is based on an extended version of PDDL for HTN planning (Castillo *et al.*, 2006), which includes a hierarchical representation of tasks and methods within a related temporal framework that allows us to specify goal deadlines, temporal landmarking between actions and a constraint propagation engine for exploiting the causal structure of plans.
- The profile adaptation is extended to offer different sequences of LOs according to the learning resource type of the LO, particularly when there is no *Requires* relation between LOs that are part of a task, as in Figure 3-1. This sequence is currently designed for the Honey-Alonso theory, but it can be easily adapted to other theories or pedagogical desires.
- The *IsPartOf* relation defined in the LO metadata is used to generate aggregation tasks or *primitive* tasks, that is, durative actions.
- The hierarchical structure is generated according to the dependency relations. If aggregated actions are completely ordered by means of a *Requires* relation, a unique method is generated. When there is an *IsBasedOn* relation with two or more actions, an auxiliary task must be created with as many methods as actions related to this LO. The associated actions have preconditions, but not their corresponding methods (see Figure 3-2,3). Finally, when a *References* relation appears, an auxiliary task is created that includes two methods: one empty with no preconditions, and another with a precondition on the student availability, as in Figure 3-4.

3.2.2 Planning Domain Definition Language-conditional domain compilation

This compilation extends the general compilation with the following features:

- It includes a full support for planning with conditional effects, which *compress* in just one operator all the effects that depend on the student learning style. For instance, in Figure 4-1

```
1-(:task Algorithms
   :parameters (?s - student)
   (:method theoretical-learning-style
    :precondition (style ?s theoretical)
    :tasks ((OR_Complexity ?s)
            (Basic-Algorithms ?s)))
   (:method pragmatic-learning-style
    :precondition (style ?s pragmatic)
    :tasks ((Basic-Algorithms ?s )
            (OR_Complexity ?s))))

2-(:durative-action Boolean-Algebra-Simulation1
   :parameters (?s - student)
   :duration (= ?duration 25)
   :condition (not
     (Boolean-Algebra-Simulation1_done ?s))
   :effect (and
     (Boolean-Algebra-Simulation1_done ?s)
     (profile-dependent effects...)))

3-(:task OR_Boolean-Algebra-Simulation
   :parameters (?s - student)
   (:method or1
    :precondition()
    :tasks (Boolean-Algebra-Simulation1 ?s))
   (:method or2
    :precondition()
    :tasks (Boolean-Algebra-Simulation2 ?s)))

4-(:task OR_Complexity
   :parameters (?s - student)
   (:method yes
    :precondition (Available ?s much)
    :tasks ((Complexity ?s)))
   (:method no
    :precondition()
    :tasks()))
```

Figure 3 Example of tasks and methods decomposition in the Hierarchical compilation

```
1-(:action Basic-Algorithms
   :parameters (?s - student)
   :precondition (and
     (not (Basic-Algorithms_done ?s)))
   :effect
     (and (Basic-Algorithms_done ?s)
          (increase (total_time ?s) 9)
          (increase (reward ?s) 5)
          (when (active ?s strong)
            (increase (reward ?s) 20))
          (when (reflective ?s strong)
            (increase (reward ?s) 40))
          (when (intuitive ?s strong)
            (increase (reward ?s) 40))
          (when (verbal ?s strong)
            (increase (reward ?s) 40))))
```

```
2-(:action OR-fictitious-Boolean-Algebra
   :parameters (?s - student)
   :precondition (and
     (not (Boolean-Algebra_done ?s))
     (Algorithms_done ?s)
     (Logic-and-Sets_done ?s)
     (Minimization-of-Circuits_done ?s)
     (Logic-Gates_done ?s)
     (or
        (Boolean-Algebra-Simulation1_done ?s)
        (Boolean-Algebra-Simulation2_done ?s)))
   :effect (and (Boolean-Algebra_done ?s)))
```

Figure 4 Example of actions in the PDDL (Planning Domain Definition Language)-conditional domain. The first action comes from a learning object (LO) with a `'Narrative Text'` learning source type (the conditional effects model the reward for each Felder dimension for this type), whereas the second uses an *IsBasedOn* relation

there are four conditional effects that depend on the values *active*, *reflective*, *intuitive* and *verbal*. This way, the operator encodes different effects or branches according to the student—the values of the rewards are computed and normalized, by giving a numeric priority, according to the work presented in Baldiris *et al.* (2008), but can be easily adapted to other pedagogical preferences. Particularly, this example corresponds to a LO whose learning source type is *Narrative Text*, the `total_time` is 9 and without *Requires* relations.

- The dependency relations (see Table 1) are generated according to the general semantics, but now a LO with an *IsBasedOn* relation generates a fictitious action, because the student only needs to follow one of the actions in the or-condition (see Figure 4-2). Thus, the corresponding LO has two *IsBasedOn* relations (*Boolean-Algebra-Simulation1* and *Boolean-Algebra-Simulation2*) and four *Requires* relations (*Algorithms, Logic-and-Sets, Minimization-of-Circuits* and *Logic-Gates*).
- It represents thresholds for each student, such as (`time_threshold ?s`) or (`reward_threshold ?s`). The former represents the total time the student devotes to the course, whereas the latter models the utility of the course for the student. This allows us to easily model rich constraints, such as (< (`total_time ?s`) (`time_threshold ?s`)).

3.2.3 Planning Domain Definition Language-temporal domain compilation

This compilation extends the general compilation with the following features:

- All actions are grounded according to the information of the students given in the problem (see Figure 5). The instantiation is done according to a matching process with the student profile. For instance, if we want to restrict a LO to be used only by a *visual* learning style, and we have two students, with *visual* and *verbal* styles, respectively, the action is only generated for the first student. The domains are now larger because we include all the applicable actions, rather than a single operator with conditional effects. But, on the contrary, it generates valid domains for planners that do not support such conditionality.
- It uses an entire numeric representation, which increases the expressiveness of the model and allows us to include metric resources and cost to model more flexible metrics, as traditionally used in P&S optimization.
- It generates both level 3 of PDDL2.1 durative actions and non-durative actions, where time is modelled by means of the artificial fluent `total_time`.
- It simulates *IsPartOf* hierarchical structures under a flat model of PDDL actions. All actions in an aggregation are enveloped within two dummy actions `Start/End` representing the beginning/ending of the aggregation (see Figure 5-1,2). `Start` contains the preconditions of the aggregation action and `End` its effects. On the other hand, the actions generated for all the aggregated actions have that `Start` as precondition (Figure 5-3). Obviously, both `Start` and `End` have duration 0.

```
1-(:durative-action Start_Discrete-Maths_Std1
   :parameters ()
   :duration (= ?duration 0)
   :condition (and
    (at start (= (Start_Discrete-Maths_Std1_done) 0))
    (at start (= (Algorithms_Std1_done) 1)))
    (at start (= (Boolean-Algebra_Std1_done) 1)))
   :effect (and
    (at end (increase (Start_Discrete-Maths_Std1_done) 1))
    (at end (increase (Discrete-Maths_Std1_done) 1))
```

```
2-(:durative-action End_Discrete-Maths_Std1
   :parameters ()
   :duration (= ?duration 0)
   :condition (and
    (at start (= (End_Discrete-Maths_Std1_done) 0))
    (at start (= (Start_Discrete-Maths_Std1_done) 1))
    all 'aggregated_actions'_Std1_done' must be 1)
   :effect (and
    (at end (increase (End_Discrete-Maths_Std1_done) 1))
    increase other numeric expressions))
```

```
3-(:durative-action Basic-Algorithms_Std1
   :parameters ()
   :duration (= ?duration 9)
   :condition (and (at start (= (Basic-Algorithms_Std1_done) 0))
                   (at start (= (Start_Algorithms_Std1_done) 1)))
   :effect (and (at end (increase (Basic-Algorithms_Std1_done) 1))
                (at end (increase (reward_Std1) 40))
                increase other numeric expressions or resource_costs))
```

Figure 5 Durative actions for student *Std1* generated for the two learning objects (LOs) of Figure 2 according to the PDDL (Planning Domain Definition Language)-temporal compilation. We assume in our problem that *Std1* is *verbal*. Since the learning resource type of `Basic-Algorithms` is `'Narrative Text'` (see Figure 2) and the student is *verbal*, this means a reward in the learning process of 40. The values of all the rewards are given by the educational experts

3.3 Problems compilation

Once the domain is generated, we compile the planning problem. The IMS-LIP standard (`http://www.imsglobal.org/profiles`) to access the student information is too wide, so we extract the relevant characteristics and compile them into a planning problem. The proposal is valid for our three approaches, although the translation differs slightly for the temporal compilation, where the problem is (interactively) generated together with the domain because of the grounded approach (Garrido *et al*., 2009).

The planning problem includes propositions to represent the objects, the initial state, the goals and the metric to optimize. The objects represent the students information for the learning design and their previous knowledge. The initial state represents the students profile, the initial values of the fluents, the language of the course and some other information (e.g. student performance, special equipment, availability, etc.). The goal is to pass the entire course or a part of it. Table 2 shows the mapping to automatically translate an IMS-LIP structure into a planning problem. The first column represents the IMS-LIP entry and the second one the corresponding item in the planning problem. As can be seen, the problem compilation is simpler than the domain compilation, as it basically consists in generating the initial values and goals for the objects.

3.4 Plans compilation

After compiling the planning domain and problem, we run a domain-independent planner to find a plan. The quality of the plan itself depends on the planner quality; some planners are optimal and return the best solution, but others return just one solution. Consequently, the domain/problem compilation has not a relevant impact in that quality. In this compilation, we translate the plan into the e-learning standard IMS-LD. Each plan represents the learning design as a route of LOs that best suits the student, which will be later displayed in a LMS. Since current LMSs support different languages, we have implemented translators for two of the most common systems: IMS-LD for dotLRN and Moodle templates. In the first case, we create a zip file that contains the input resources (LOs) as well as the learning design as an XML file. Our algorithm compiles the next six items in the XML file:

- Goals, which are taken from the goals of the planning problem, usually the effects of the final action of any domain.

Table 2 IMS-LIP mapping to a problem compilation—irrelevant information has been ignored

IMS-LIP →	Planning problem
identification/name/contentype/ referential/indexid	:objects student-name, checking its uniqueness
accessibility/preference/typename/ typevalue/Learner_Style_Processing/ prefcode/profile-type.value	:inits (profile-type student-name value) e.g. :inits (reflective student1 strong)
goal/typename/typevalue/course-name/ contentype/temporal/typename/ time_threshold/temporalfield/value	:inits (= (time_threshold student-name) value) :goals (= (course-name_done student-name)) e.g. :inits (= (time_threshold student1) 3800) e.g. :goals (course-name_done student1)
activity/typename/typevalue/task/ learningactivityref/text/object-name	:objects object-name :inits (known student-name object-name) e.g. :objects graph_theory e.g. :inits (known student1 graph_theory)
accessibility/language/typename/ typevalue/lang-value	:inits (language_level lang-value student-name high) e.g. :inits (language_level English student1 high)
competency/contentype/referential/ indexid/performance-level/ description/short/value	:inits (performance_level student-name high) e.g. :inits (performance_level student1 high)

IMS-LIP = IMS Learner Information Package.

- Prerequisites, which are taken from the initial conditions on previous knowledge that is required to follow this course. They are the links to LOs or objectives of other courses, which allow us to perform multiple course planning in the future.
- Roles. In this case, the only role is the student for whom the learning design is generated.
- Activities, which are taken from the plan. Iterating for each action, an IMS-LD activity entry is generated, which also includes a link to the corresponding LO. Obviously, all fictitious actions used during planning are now omitted.
- Activity structure, which relates to the plan itself and its route of actions. Given that the IMS-LD standard allows other control structures, such as conditional plans with branches, in the future we will study how to generate conditional plans and the impact it has on the fact that students may execute different alternatives.
- Resources. For each LO in the input IMS-MD, a resource, that can or not be used in the plan, is defined.

The second translator compiles the plan into a Moodle template that describes an XML file, which is related to the course previously implemented in Moodle. The compilation algorithm is straightforward, as the XML document simply contains the student identifier and an enumeration of the LOs to be executed. The sequence in which LOs appear in the document corresponds to the particular ordering for each student, which helps the student explore and navigate through the course.

3.5 Approaches comparison

Table 3 shows the differences between the three compilations from a knowledge engineering perspective based on several characteristics. Although the underlying semantics in the three compilations is the same, each compilation adapts more naturally to each learning language. More specifically, the *Hierarchical* compilation adapts better to the Moodle format in terms of domain and problem, whereas the *Conditional* one adapts better to the IMS-MD standard. This is because of the nature of the Moodle courses, which usually focus more on the inner structure of the course,

Table 3 Comparison of the three compilations

Characteristic	Hierarchical	PDDL-conditional	PDDL-temporal
Domain definition	Moodle	IMS-MD	Both
Problem definition	Moodle and IMS-LIP	IMS-LIP	Interactively
Goal definition	IMS-LIP	LO in MD and IMS-LIP	Problem
Deadline definition	IMS-LIP	IMS-LIP	Problem
Prerequisite definition	IMS-LIP	IMS-LIP	Problem
Dependency relations	All	IsBasedOn Requires	All
Students profile	Honey-Alonso	Felder	Both
Soft preconditions	Tasks and methods	Domain	Domain (using References)
Time management	Durative actions	Fluent	Both
Metric	No	Yes	Yes
Planner	SIADEX	Conditional effects Metrics	Temporal Metrics
Plan compilation	Moodle/IMS-LD	Moodle/IMS-LD	Moodle/IMS-LD

PDDL = Planning Domain Definition Language; IMS-MD = IMS Meta-Data; IMS-LIP = IMS Learner Information Package; IMS-LD = IMS learning design; LO = learning object.

rather than IMS-MD, which focuses more on the definition of particular LOs and leaves apart somewhat the relations between LOs. On the other hand, the *Temporal* compilation accepts both formats, as it manages both hierarchical and flat structures. The planning goals, deadlines (time limit devoted to the course) and prerequisites (student's previous knowledge for the course) are defined in the IMS-LIP for the *Hierarchical* and *Conditional* approaches, whereas the *Temporal* compilation uses the data defined in the problem. The row *Dependency* relations considers the types of relations supported by the different approaches. *Students profiles* refer to the styles that are currently supported, but this can be easily extended. *Soft preconditions* refer to the fact that the learning design can contain LOs that, without being mandatory, provide some benefit to the student. This is possible through the methods in the *Hierarchical* approach and through the precondition (< (total_time ?s) (time_threshold ?s)), which allows the *Conditional* representation to include more LOs than strictly necessary. *Time management* represents how the approaches deal with time. The row *Metrics* means whether the approach can manage quality metrics or not. The row *Planner* represents the planner required to solve the problems modelled by each compilation. The last row shows how the plans, which the corresponding planner generates, comply with Moodle and IMS-LD.

In summary, the *Hierarchical* compilation permits modelling many e-learning features, but only the planner SIADEX (Castillo *et al*., 2006) supports them, and it does not support quality metrics yet. The other compilations use PDDL and deal with optimization metrics, such as maximizing the learning reward. However, most state-of-the-art planners cannot maximize metrics, so a metric for maximizing the total utility that the LOs report to the student cannot be always used and represents an important challenge for existing planners. From a pedagogical side, it is not easy (nor possible) to transform the metric for maximizing the reward into a metric for minimizing its inverse, due to how rewards are computed in this domain. In particular, we use a mapping defined in Baldiris *et al*. (2008) that contains information on whether the learning resource type is *good*, *very good* or *indifferent* for each learning style. But we cannot assert anything about the inverse. For instance, it is untrue that if a *lecture* is *very good* for a reflective student it has to be *very bad* for a non-reflective student. So, it is unclear which values to assign to those cases not covered in Baldiris *et al*. (2008).

The main conclusion is that there is not clearly a best approach. In some cases, the hierarchical compilation shows more natural (particularly when there are many *IsPartOf* relations); in others, the conditional compilation is very appealing to model and subsume all alternative branches that

students may follow; and in other cases, the temporal compilation shows more expressive, because of its simulation of hierarchical structures and use of a complete numeric representation.

4 Experimental results

In this section, we perform a quantitative and qualitative evaluation. From a quantitative perspective, we are interested in technical experiments to assess the correctness of the compilation schemas from a planning perspective. Thus, we have run some experiments with four courses defined in IMS-MD. The first one is a complete AI course containing LOs for covering all the typical tasks, such as reading a subject, practising, programming, performing exams, etc. Each task includes from one to six optional ones with different degrees of adequacy to the learning styles. The course has 172 LOs, 32 of them representing common tasks and 14 hierarchical aggregations. This course contains enough LOs for testing the planning complexity and for covering realistic-size courses. The other courses, `Representation, Planning` and `Search`, are subsets of this.

After compiling the IMS-MD course following the three approaches, we have defined the corresponding planning problems. All the compilation processes finished in a few seconds. We have used one planner for each compilation: SIADEX for the hierarchical domain, CBP (Fuentetaja *et al.*, 2009) for the *Conditional* domain and LRNPLANNER (Garrido & Onaindía, 2010) for the *Temporal* domain. We have also analyzed the applicability of many state-of-the-art planners that participated in the ICAPS planning competitions (`http://ipc.icaps-conference.org`), but they do not support all the domain requirements at the same time, that is, conditional effects, `or` and negative preconditions, fluents and cost metrics. Therefore, we have used these three planners because they support all the domain requirements for each compilation and are sound (all their generated plans are valid). SIADEX is a knowledge-based HTN planner with temporal features. CBP includes the original algorithms of METRIC-FF, and it also implements new heuristics and algorithms to better deal with cost metrics—the original METRIC-FF could not solve all the tested problems. LRNPLANNER deals with durative actions and metrics. We also tried to solve the durative problems with MIPS-XXL but it has scalability problems due to the numeric representation; all the information is encoded as numeric functions and current planners have problems with this kind of reasoning.

First, we performed some experiments to analyze the scalability of the approaches. We considered problems with 1, 2, 4, 8, 16, 32 and 64 students and measured the running time for solving the problems. For the longest course (AI course), the time to solve the problems ranged: in the *Hierarchical* approach from 0.04 s for 1 student up to 97.41 s for the 64 students; in the *Conditional* one from 0.47 s to 455.94 s; and in the *Temporal* one from 0.07 s to 37.29 s. The execution time for the other domains were proportional to the course size and number of students (we do not show detailed results for lack of space). This means that all the approaches can cope well with a reasonable number of students. The size of the plans ranged progressively: in the *Hierarchical* approach from 86, 172 (86*2), 258 (86*3)… 5504 (86*64) actions for 1, 2, 3… 64 students, respectively; in the *Conditional* one 78, 156 (78*2)… 4992 (78*64) actions; and in the *Temporal* one from 66 to 4224 (66*64) actions.

Second, we performed other experiments to test the quality of the plans, that is, e-learning designs. Obviously, quality depends on the pedagogical theory used and how it is defined by the education experts. In our approach, the *Hierarchical* domain is based on the order in which the LOs are presented to each student, while the *Conditional* and *Temporal* domains are based on the utility (reward) theory mentioned above. There are very specific courses with evaluations based on a student satisfaction-oriented perspective (Castillo *et al.*, 2010). But our tested courses are more generally designed and use a theory that implies the planner should find a plan that maximizes the total reward without exceeding the total time the student can devote to the course. That is, the solution plan must contain as many LOs that fit the particular student profile as the time constraint allows. For example, the plan for an *active* student should contain LOs that the recommendation table in Baldiris *et al.* (2008) suggests for active students. However, as current state-of-the-art planners cannot maximize metrics, we have transformed the reward maximization into a *penalty*

minimization metric. As mentioned before, there is not an exact transformation of metric for maximizing the reward into a metric for minimizing its inverse, so we use an approximation. The recommendation table contains information on whether the learning resource type is *good*, *very good* or *indifferent* for each learning style. We use a *penalty* PDDL-fluent that increases in two units each time the plan includes an *indifferent* action, in one unit when it includes a *good* action, and there is not penalty for *very good* actions, but this can be easily modified to other values. This way, we guide the planner to include *very good* actions in the plan, that is, actions that provide greater reward to the student. We vary the domain and the student profile, and measure: (i) the size, that is, number of LOs in the solution plan, (ii) the total time and penalty of the learning design and (iii) the differences between plans. For these differences, we take the first plan (the one for the active student) as the base plan and count the number of LOs in the other plans that are not included in it as a measure of adaptation to students. The bigger the difference is, the more actions are found for that particular type of student that are not suggested in the baseline. In the *Hierarchical* approach, the differences are the number of LOs that are presented to the student in a different order than the base plan. We consider problems with the four profiles in the Honey-Alonso taxonomy, that is, active, reflexive, theoretical and pragmatic, which correspond to the active, reflective, intuitive and sensitive ones in the Felder taxonomy, respectively. Table 4 shows the results. The rows represent the planning problems: student profile (P: 1-active, 2-theoretical/intuitive, 3-pragmatic/sensitive and 4-reflective) and courses (the first four rows correspond to the Representation course, the following four rows to the Planning course, the following rows to the Search course and the last ones to the complete AI course). Columns represent: number of LOs in the plan (LOs), total time (Ti) of the course, penalty (Pe) and number of differences (Di). The *Conditional* approach uses an additional reward threshold; a plan is valid only if the total reward is equal or greater than this threshold, which poses the question of setting the initial reward threshold value. We have initially set it to zero and executed the planner. The reward obtained in the solution plan is the threshold value used in the experiments. As the base case we used the solution returned by CBP without using any metric nor reward threshold.

Table 4 Experimental results for the adaptation of plans

	Base case				PDDL-conditional				PDDL-temporal				HTN-PDDL		
P	LOs	Ti	Pe	Di	LOs	Ti	Pe	Di	LOs	Ti	Pe	Di	LOs	Ti	Di
1	15	650	10		29	825	6		20	755	7		14	650	
2	15	650	7	0	30	935	10	7	20	815	10	2	14	650	0
3	15	650	16	0	28	735	11	1	20	755	11	0	14	650	
4	15	650	7	0	30	845	10	6	20	755	11	0	14	650	0
1	13	510	7		26	715	1		12	570	1		13	600	
2	13	510	4	0	18	660	3	10	12	420	3	3	13	600	13
3	13	510	10	0	27	625	2	2	12	570	12	0	13	600	
4	13	510	4	0	18	720	4	8	12	510	4	2	13	600	13
1	40	1430	29		49	1785	17		40	1730	12		44	1620	
2	40	1430	14	0	51	1685	14	11	40	1430	14	9	44	1620	31
3	40	1430	34	0	46	1455	19	10	40	1620	14	4	44	1620	
4	40	1430	17	0	48	1595	15	10	40	1430	17	9	44	1620	31
1	66	2530	45		81	2925	22		66	2655	26		86	3290	
2	66	2530	28	0	87	2845	27	23	66	2335	29	14	86	3290	55
3	66	2530	59	0	80	2595	34	11	66	2705	37	2	86	3290	
4	66	2530	32	0	84	2815	30	18	66	2215	38	14	86	3290	55

PDDL = Planning Domain Definition Language; HTN = Hierarchical Task Network; LOs = learning objects; Ti = total time of the course; Pe = penalty; Di = number of differences.
Running time was <0.2 s in all cases.

Table 5 Questionnaire for a qualitative evaluation of learning designs by nine lecturers

	Strongly disagree			Disagree			Neutral			Agree			Strongly agree		
Question	C	T	H	C	T	H	C	T	H	C	T	H	C	T	H
Q1				2	1		2	3	4	4	4	4	1	1	1
Q2				2	1		4	5	5	2	2	4	1	1	
Q3		1		1			7	7	5	1		4		1	
Q1				2	2		2	4	4	4	2	4	1	1	1
Q2				1	3		6	4	5	1	2	4	1		
Q3		1		1	1		7	6	3	1		6		1	

PDDL = Planning Domain Definition Language; HTN = Hierarchical Task Network; LOs = learning objects.
C, T and H stand for PDDL-conditional, PDDL-temporal and HTN-PDDL compilations, respectively.
Q1. The number of LOs is appropriate for the course.
Q2. The duration of the LOs sequence is appropriate.
Q3. The adaptation of contents to the students learning style is appropriate.

The total time thresholds (maximum time the student can devote to the course) were bigger enough, that is, there were not time constraints.

The results show that the plans generated with the *Conditional* and *Temporal* approaches better adapt to the student profile. Even though there are fewer LOs in the base plans than in the others, the penalty is nearly always greater. That means that our approaches find learning designs with a higher number of LOs, that is, the student has more possibilities to better learn the course, and most of them are *very good* for its profile. The *Hierarchical* approach also shows adaptation to the student profile according to the theory it is based on: it only distinguishes two kinds of students, pragmatic and theoretical ones (the active style is considered as pragmatic and reflective as theoretical). The plans generated for both kind of students contain several actions in different order, but the first domain is too simple to appreciate this difference.

Finally, in order to verify that the e-learning designs obtained by applying the proposed methodology are valid, that is, correctly specified, we have compiled the plans into the e-learning standard IMS-LD and we have correctly uploaded the resulting XML files to the learning dotLRN (`www.dotlrn.org`) and RELOAD (`http://www.reload.ac.uk`) platforms.

On the other hand, a thorough qualitative evaluation of the approaches is difficult, as it involves the collaboration of educational researchers for a correct definition of the courses and for a comprehensive testing with real students. The work presented in Castillo *et al.* (2010) reports some preliminary experiments in this direction for the *Hierarchical* compilation. The main result here is that it is not easy to measure the goodness of our approach via the students grades/scores, as following a learning design does not necessarily mean achieving a better grade. Therefore, we have designed a brief questionnaire to evaluate the quality of the learning designs, that is, plans for the whole AI course detailed above, and their adequacy to the students' profiles from the lecturer perspective. Table 5 shows the results of this evaluation by nine lecturers of AI courses, in terms of the three different compilations and two different learning styles. In particular, the first three rows (questions) refer to the votes for the plans obtained for active/pragmatic students, whereas the following three are for the reflective/theoretical ones. In general, the experts agree (though not very strongly) with the designs given by the three compilations in terms of their form, size and adaptation to the students. The main reason why there were not more strongly agree answers was that we had loosened too much the constraints (relations) among LOs, thus allowing the planners to generate plans with non-standard orders in the sequence of activities. Even so, the generated orders were correct with respect to the initial LOs. Also, an additional test with respect to the compilation with the most coherent and consistent sequence of LOs shows that one lecturer

prefers the conditional compilation, three the temporal conditional and five the HTN one, as it seems more natural. Finally, and according to the lecturers opinion, it is extremely hard to design (and evaluate) an adequate learning design to each profile, but they agree with the usefulness of applying planning technology to help their labor.

5 Conclusions and future work

This paper has contributed with both a general and three instantiated approaches for automatic compilations that interpret and translate e-learning models to planning. A course definition is represented as a planning domain, the student learning information as a planning problem for that domain and the learning design as the plan generated by a domain-independent planner. Our three different compilations of planning domains and problems allow current planners to automatically generate valid learning designs. Our experimental results have raised challenges and shown scalability limitations in some cases. Also, a complete analysis of our compilations has allowed us to detect three important drawbacks that show *semantic gaps* between the pedagogical decisions and the automated translations. First, instructional designs are complex to model; they tend to be theory-independent and do not capture the pedagogical knowledge required to generate a course. Second, e-learning languages are usually too generic and try to cover too many aspects, making the implementation of general and suitable translators for all LMSs very difficult and, in some cases, slightly imprecise (e.g. the learning resource types). Third, despite the expressive power of e-learning languages, there are still some aspects that cannot be represented and are essential in P&S. For example, the definition of the resources involved in the tasks, their costs, the temporal constraints on availability and how these resources are to be managed are important lacks in e-learning languages. In other words, e-learning standards still miss some information for pedagogically interesting planning. Analogously, a more flexible approach for requiring/achieving the learning outcomes is also missing. For instance, the execution of a task may result in a higher effect in one student than in another, depending on the student learning style. Informally, two students do not learn the same with the same task, and the definition of this type of conditional effects should be available in the e-learning standards. We have solved this problem by implementing pedagogical suggestions into the compilers. But all in all, our compilations show very appropriate for integrating e-learning standards and AI planning, while also providing the strong foundations to be extended to other particular education theories.

In the future, we intend to overcome the previous drawbacks by addressing three parallel lines. First, coming up with more expressive models of actions for planning e-learning activities, thus augmenting the semantics of the model, which will increase the opportunities to: (i) deal with complex course composition, and (ii) validate and resolve courses with similar but incommensurate LOs. Second, extending our system to assist the course designer in making sure that the same naming conventions are used by adopting a standard or common ontology, as proposed in Kontopoulos *et al.* (2008), thus avoiding unsolvable situations (e.g. having a LO requiring knowledge on *planners* and a student knowing about *planning systems*). Third, working on the actual execution of courses and the dynamic planning adaptation of courses with respect to the real behavior of students.

Acknowledgements

This work has been supported by the Spanish MICINN under projects TIN2008-06701-C03 and Consolider Ingenio 2010 CSD2007-00022, by the Mexican National Council of Science and Technology and the regional projects CCG08-UC3M/TIC-4141 and Prometeo GVA 2008/051.

References

Alonso, C. & Honey, P. 2002. *Honey-alonso Learning Style Theoretical Basis* (in Spanish). Retrieved December 2012, from `http://www.estilosdeaprendizaje.es/menuprinc2.htm`.

Baldiris, S., Santos, O., Barrera, C., Boticario, J.G., Velez, J. & Fabregat, R. 2008. Integration of educational specifications and standards to support adaptive learning scenarios in ADAPTAPlan. *Special Issue on New Trends on AI Techniques for Educational Technologies. International Journal of Computer Science and Applications* **5**(1), 88–107.

Boticario, J. & Santos, O. 2007. A dynamic assistance approach to support the development and modelling of adaptive learning scenarion based on educational standards. In *Proceedings of Workshop on Authoring of Adaptive and Adaptable Hypermedia, International Conference on User Modelling*, Corfu, Greece, 1–8.

Camacho, D., R-Moreno, M.D. & Obieta, U. 2007. CAMOU: a simple integrated e-learning and planning techniques tool. In *4th International Workshop on Constraints and Language Processing*, Roskilde University, Denmark, 1–11.

Castillo, L., Fdez.-Olivares, J., García-Perez, O. & Palao F. 2006. Efficiently handling temporal knowledge in an HTN planner. In *Proceedings of 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, Borrajo, D. & McCluskey, L. (eds.). AAAI, 63–72.

Castillo, L., Morales, L., Gonzalez-Ferrer, A., Fdez-Olivares, J., Borrajo, D. & Onaindia, E. 2010. Automatic generation of temporal planning domains for e-learning problems. *Journal of Scheduling* **13**(4), 347–362.

Essalmi, F., Ayed, L.J.B., Jemni, M., Kinshuk, & Graf, S. 2010. A fully personalization strategy of E-learning scenarios. *Computers in Human Behavior* **26**(4), 581–591.

Felder, R. M. 1996. Matters of style. *American Society for Engineering Education Prism* **6**(4), 18–23.

Fox, M. & Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* **20**, 61–124.

Fuentetaja, R., Borrajo, D. & Linares López, C. 2009. A look-ahead B&B search for cost-based planning. In *Proceedings of CAEPIA'09*, Murcia, Spain, 105–114.

Garrido, A. & Onaindía, E. 2010. On the application of planning and scheduling techniques to E-learning. In *Proceedings of the 23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2010)*—Lecture Notes in Computer Science **6096**, 244–253. Springer.

Garrido, A., Onaindia, E., Morales, L., Castillo, L., Fernandez, S. & Borrajo, D. 2009. Modeling E-learning activities in automated planning. In *Proceedings of the 3rd International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS-2009)*, Thessaloniki, Greece, 18–27.

IMSLD 2003. *IMS Learning Design Specification. Version 1.0 (February, 2003)*. Retrieved December, 2012, from `http://www.imsglobal.org/learningdesign`.

IMSMD 2003. *IMS Learning Resource Meta-data Specification. Version 1.3 (August, 2006)*. Retrieved December, 2012, from `http://www.imsglobal.org/metadata`.

Kontopoulos, E., Vrakas, D., Kokkoras, F., Bassiliades, N. & Vlahavas, I. 2008. An ontology-based planning system for e-course generation. *Expert Systems with Applications* **35**, 398–406.

Limongelli, C., Sciarrone, F. & Vaste, G. 2008. LS-plan: an effective combination of dynamic courseware generation and learning styles in web-based education. In *Adaptive Hypermedia and Adaptive Web-Based Systems, 5th International Conference, AH 2008*, Nejdl, W., Kay, J., Pu, P. & Herder, E. (eds.)., 133–142. Springer.

Méndez, N.D.D., Ramírez, C.J. & Luna, J.A.G. 2005. IA planning for automatic generation of customized virtual courses. *Frontiers in Artificial Intelligence and Applications. Planning, scheduling and constraint satisfaction: from theory to practice* **117**, 139–148.

Mohan, P., Greer, J. & McCalla, G. 2003. Instructional planning with learning objects. In *IJCAI-03 Workshop Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, Mexico, 52–58.

Sharable Content Object Reference Model (SCORM) 2004. Retrieved December, 2012, from `http://scorm.com`.

Sicilia, M.A., Sánchez-Alonso, S. & García-Barriocanal, E. 2006. On supporting the process of learning design through planners. *CEUR Workshop Proceedings: Virtual Campus 2006 Post-Proceedings*. Barcelona, Spain, **186**(1), 81–89.

Ullrich, C 2008. Pedagogically founded courseware generation for web-based learning, No. 5260, Lecture Notes in Artificial Intelligence **5260**, Springer.

Ullrich, C. & Melis, E. 2009. Pedagogically founded courseware generation based on HTN-planning. *Expert Systems With Applications* **36**(5), 9319–9332.