Modeling E-Learning Activities in Automated Planning *

Antonio Garrido and **Eva Onaindia**
Universidad Politecnica de Valencia.
Camino de Vera s/n, 46071 (Valencia). Spain

Lluvia Morales and **Luis Castillo**
Universidad de Granada.
Granada. Spain

Susana Fernández and **Daniel Borrajo**
Universidad Carlos III de Madrid.
Leganés (Madrid). Spain

Abstract

This paper presents three approaches to generate learning designs using existing domain-independent planners. All of the approaches compile a course defined in a standard e-learning language into a planning domain, and a file containing student's learning information into a planning problem. The learning designs are automatically generated from the plans that solve the problems. The approaches differ in the kind of planning domain generated, thus increasing the possibilities of using existing planners: i) hierarchical, ii) including PDDL actions with conditional effects, and iii) including PDDL durative actions. We also analyse the pros and cons on the knowledge engineering procedures used in each approach.

Introduction

Sequencing of learning activities, according to different student's profiles and pedagogical theories, has been a widely studied subject by planning community for at least a decade (Brusilovsky & Vassileva 2003; Castillo *et al.* 2009; Vrakas *et al.* 2007). This sequencing depends on temporal conditions given by the needs of each student, the course duration, the available resources and even collaboration between tutors and students, which makes the problem very interesting for the AI P&S community.

However, acquiring enough information on educational domains to represent them as a planning domain is not an easy work at practice. Until few years ago, there was no standard language to represent most of the many aspects involved in learning activities sequencing. Thanks to the recent rise and widespread use of specification languages based on XML schemata, such as IMS-MD, IMS-LIP, and IMS-LD (IMS-GLC 2001 2009), the e-learning community can now represent information on educational domains in full detail. Despite this, designing generic translators from the information of those standards into a planning domain representation can be difficult because of two main reasons: i) people give different meanings and uses to the fields of the standards, given that the standards provide some flexibility on how to represent knowledge, and ii) there is a great variety of planning paradigms.

*This work has been partially supported by the Spanish MICINN under projects TIN2008-06701-C03-03 and TIN2005-08945-C06-05, and the regional project CCG08-UC3M/TIC-4141.

This paper focuses on how to model learning scenarios (learning objects and students) as planning domains+problems, and on the automated translation from standard e-learning languages to planning models. More particularly, the paper presents a general architecture which consists of three translation approaches to compile learning designs, based on IMS-MD and IMS-LIP standards (IMS-GLC 2001 2009), into planning domains, as depicted in Figure 1. These domains, together with the file compilations that contain students' learning information, are later solved by existing domain-independent planners. Thus, the resulting plans represent tailored sequences of learning activities that students must follow. And this represents an important advantage: each learning design comprises a personalised plan that fully fits each student's necessities and preferences, learning styles and lets him/her work at his/her own pace. Finally, the plan is translated into another standard representation, called IMS-LD, that displays the learning design on different on-line learning platforms. In essence, this paper contributes with:

- An automated translation of IMS-MD and IMS-LIP e-learning templates into three different planning compilations: i) hierarchical, ii) PDDL-conditional, and iii) PDDL-temporal.
- An intuitive graphic tool that enriches the metadata labelling of the learning objects. This enrichment plays an important role, as it serves to complete information that is not always described from an educational point of view, but still needed for AI planning.
- An effective use of planning technology to generate learning designs that best suit students' learning goals, thus promoting a more personalised access to the learning objects.
- An additional translator that parses the resulting plans, and generates the input resources (learning objects) and learning design to be included in state-of-the-art learning platforms, such as dotLRN and Moodle, thus closing the e-learning cycle.

The paper is organised as follows. First, we briefly describe the e-learning basis on which our work is based. Next, we include some related work. After that, we analyse how to model learning designs in planning, and present the translation section with the templates used to convert e-learning

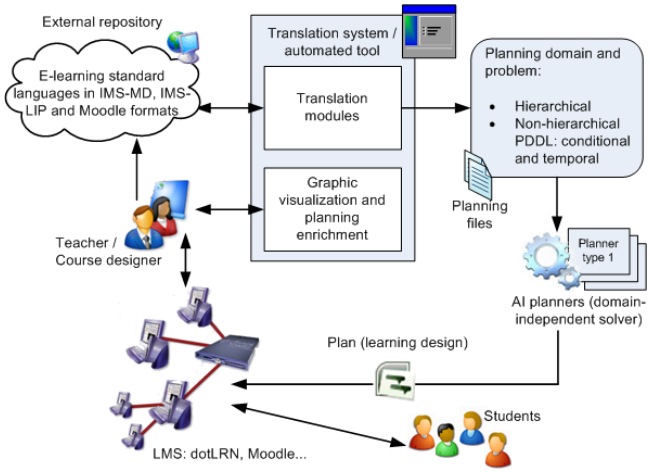

Figure 1: Overview of our system's architecture.

standards into planning domains. Then, we introduce our graphic tool that allows us to enrich the courses from a planning standpoint. Later, we discuss the options to use the resulting plans. Finally, we conclude the paper and motivate the future work.

Basic Background on E-Learning

There are many standards that the e-learning community uses to deal with learning activities and their sequencing. The most famous one is SCORM (Wisher 2009) which integrates some of the IMS (IMS-GLC 2001 2009) familiy standards but not those that model student profiles and sequencing personalization. The IMS family standards are composed, among others of the following languages:

- IMS-MD, which is responsible for describing learning activities and relations among them. It uses an XML schema with nested labels such as *title*, *resourceType*, *learningTime*, etc.
- IMS-LIP, which integrates every student's profile data in a single document. The standard has also nested labels; e.g. *identifier*, *name*, *preferences*, *competencies*, etc.
- IMS-LD, which includes the relations between information and sequencing of learning activities, and one or several students' profiles.

These standards are supported by some learning management systems (LMS). We have established an XML-RPC and SOAP communication protocol with two of them, Moodle and dotLRN respectively, capable to retrieve and provide information about them in order to be used in the real world. Moodle supports IMS-MD and IMS-LIP standards and has an API that returns them in a special format shown in Figure 2. dotLRN can additionally display learning activities sequences using the IMS-LD standard.

The first two standards are relatively easy to use and that is the reason why its extensive spread over the past years. However, despite the IMS-LD's first stable version is already six years old, it has not been widely accepted in the


```
- <row>
    <Identifier>ITEM-task3</Identifier>
    <Title>Algorithms</Title>
    <language />
    <coverage />
    <otherPlatformRequirements />
    <learningResourceType />
    <difficulty />
    <typicalLearningTime />
    <resource />
    <resourceId />
    <honeyAlonsoStyle />
  - <isPartOf>
      <IPO0>ORG-s2ctest2</IPO0>
    </isPartOf>
    <isBasedOn />
  - <requires>
      <REQ0>ITEM-action6</REQ0>
      <REQ1>ITEM-task5</REQ1>
    </requires>
    <references />
  </row>

- <row>
    <Identifier>ITEM-action6</Identifier>
    <Title>Basic Algorithms Lecture</Title>
    <language>es</language>
    <coverage>Mandatory</coverage>
    <otherPlatformRequirements>NoOne</otherPlatformRequirements>
    <learningResourceType>Narrative Text</learningResourceType>
    <difficulty>very easy</difficulty>
    <typicalLearningTime>9</typicalLearningTime>
    <resource>test-src/one.html</resource>
    <resourceId>RES-2DB0D82CFCB32EBC9B9D961486EC04A6</resourceId>
    <honeyAlonsoStyle />
  - <isPartOf>
      <IPO0>ITEM-task3</IPO0>
    </isPartOf>
    <isBasedOn />
    <requires />
    <references />
  </row>
```


Figure 2: Two learning objects of an XML course in Moodle format.

e-learning community. It is really difficult for one person to design a plan that considers most of the variables to sequence learning activities in a course and adapt that sequence to every student's profile. And this is the main motivation for using P&S techniques, as they can be very powerful to automatically generate IMS-LD documents containing sequences of learning activities fully adapted to the students.

Related Work

The application of integrated P&S techniques to improve the sequencing of learning activities from Intelligent Tutoring Systems was first introduced in (Peachy & McCalla 1986). The underlying idea for using P&S techniques is to deal with causal (planning) and temporal+resource constraints (scheduling) capabilities in an e-learning setting. Since then, some works have appeared but proposing *ad-hoc* approaches that do not consider standard labellings of learning objects. In (Mohan, Greer, & McCalla 2003), an exhaustive IMS-MD standard labelling over learning objects for the definition of a domain model was proposed, which acted as a precursor for more standard approaches.

The approach proposed in (Camacho, R-Moreno, & Obieta 2007) works over e-learning courses and adapts the plan sequencing to the IMS-LD standard. However, it does not take advantage of the entire standards for domain modelling. Also, it does not support temporal constraints on particular learning activities, but on the entire course. The PASER system (Kontopoulos *et al.* 2008) uses ontologies over learning activities curricula, after the planning process, to simulate the *lack* of information about its causal relations. However, that system does not support scheduling features, such as temporal reasoning, or adaptation capabilities to learning styles and tutor interaction. On the other hand, the approach presented in (Ullrich & Melis 2009) combines both IMS-MD metadata for domain modelling and a set of competencies required by the students. This approach generates an activities sequence over a domain-dependent intelligent tutoring system, which unfortunately makes it lose the ability to generalise the translation process and a further application of its results.

The work presented in this paper shares the ideas on e-learning standard usage and generation of sequence of activities. But, we address that by using a knowledge engineer-

ing algorithm to directly generate the planning domain of a course in a standard Planning Domain Definition Language (PDDL), so that we can easily use state-of-the-art planners to solve the problem.

Modelling Learning Design in Planning

Learning designs keep a strong resemblance with AI planning models. After all, they both rely on the underlying idea of using a sequence of activities that are *linked* by cause-effect relations and make it possible to achieve some (learning) goals. From a general perspective, most learning designs share the following features:

- A course is defined by a set of different learning activities, also known as *learning objects*. Usually, they are represented as XML schemata (see Figure 2). For example, in the definition of an AI course, there may be a generic task for reading the introduction to planning. And, there could be several learning objects to accomplish it, such as viewing a slide presentation, reading the introduction text from the text book, searching for the concept of Automated Planning in the Web and reading a couple of pages, or seeing a graph about planning. It is enough for the student to follow any one of them to accomplish the task, but more than one can be performed as well. This set of options is usually called the *Metadata (MD)* set.
- Each activity can be more or less appropriate to each student depending on the student's profile. Therefore, we need mechanisms to determine this profile. There are many theories that classify students into a set of profiles. Two well-known examples are the Felder's learning styles[1] (Felder 1996) and the Honey-Alonso ones (HoneyAlonso 2002). These learning styles can be translated as order rules or utilities according to the planning paradigm used.
- Learning activities can have dependency relations among them. For instance, before reading about PDDL, the student should have some knowledge on predicate logic. More formally, we support four types of relations that include hierarchical structures and ordering relations based on content dependencies. The hierarchical structures use the *IsPartOf* IMS-MD relation, which represents a hierarchical aggregation of learning objects. Additionally, there are three types of causal dependencies, *Requires*, *IsBasedOn* and *References*, as in LOM terminology (LOM 2002). We interpret the first two relations as hard requirements. In the case of the Required elements, all of them have to be completed before initiating a new learning object. Let us assume that 'A Requires B' and 'A Requires C'. In this case, B and C need to be finished before doing A. In the case of the IsBasedOn elements, at least one of them has to be completed. Assuming 'A IsBasedOn B' and 'A IsBasedOn C', only B or C must be completed before initiating A. Intuitively, the Requires and IsBasedOn relations represent the idea of conjunctive and disjunctive requirements. On the other hand, the course designer might also recommend other previous objects by means of the References relation. This relation does not denote a hard requirement but a recommendation (soft requirement) to complete a learning object before proceeding with the next one.
- Each activity takes a standard time (duration) to fulfill, commonly known as *typical learning time*. In Figure 2, however, ITEM-task3 has no duration (typicalLearningTime) as it is derived from its aggregated learning objects. In other words, duration is only specified for *primitive* activities.
- Each learning activity belongs to a source type, such as a lecture, a diagram, an exercise, etc. Although this does not seem to be very relevant for planning, it really has a positive or negative impact in the outcome of the activity. According to education experts, the source type highly interacts with the student's profile. For instance, a lecture is very recommendable for Felder's verbal students but not for visual ones, and just the opposite holds for a diagram.
- The *Metadata* set of learning objects is translated into a planning domain where each learning object is represented as one or several planning actions. As a general rule, we can state that one planning domain is defined per course, but we can use course composition and create more general and larger domains, thus providing more opportunities to reuse the learning objects in other contexts. The planning problem comprises one or more students, where the students' profiles and initial background are encoded as propositions included in the initial state. Finally, the goals usually consist in attaining some levels of knowledge in particular topics or even in accomplishing the whole course.

It is important to note that this information is part of the IMS standard and not specially introduced for AI planning. But, to the specific purpose of using planning techniques, it is essential not to have missing components and to deal always with coherent information. For instance, a situation like 'A Requires B' and 'B Requires A' would entail a failure to find a plan[2].

As seen from above, learning objects, with their duration, student profile's dependence and the relations defined in their metadata can be metaphorically considered as traditional actions used in AI planning domains. In particular, each learning object can be simply modelled as an action, its dependency relations as preconditions, and its outcomes as effects. For instance, the learning objects of Figure 2 could be modelled by means of a PDDL-like structure similar to the next one:

[1]The learning styles model developed by Richard Felder incorporates four dimensions: the Perception dimension (sensitive/intuitive), the Processing dimension (active/reflective), the Input dimensions (visual/verbal) and the Understanding dimension (sequential/global).

[2]The tool that we present below allows the user to visually notice this situation and, therefore, helps validate metadata labelling.

```
(:action ITEM-task3 ;; Algorithms
   :parameters (?s - student)
   :duration ... ;; defined by its aggregated actions
   :precondition (and (ITEM-action6)
                      (ITEM-task5)
                      student's profile requirements...)
   :effect (and (ITEM-task3-done)
                other student's profile-dependent effects...)
)

(:action ITEM-action6 ;; Basic Algorithms Lecture
   :parameters (?s - student)
   :duration 9
   :precondition (student's profile requirements...)
   :effect (and (ITEM-action6-done)
                other student's profile-dependent effects...)
)
```

The basic elements of an action, such as preconditions, duration and effects, are easily recovered from each learning object's metadata and generated in the planning translation. However, there are other elements that are not easy nor intuitive, such as the *IsPartOf* (hierarchical structure) or *References* (soft preconditions) relations, the conditionality/interaction that appears when dealing with different source types and students' profiles, etc. All these elements impose important challenges during the planning compilation, which are more or less significant depending on the planning approach to be used. This reflects the need of emphasising the knowledge engineering methods to perform such compilations.

Translators

This section describes in detail the three different compilations to be subsequently used by domain-independent planners. The most intuitive approach is the *hierarchical* one, where a learning design is modelled as a task hierarchy containing durative actions. Next one is the *PDDL-conditional* that includes PDDL actions with conditional effects and, finally, the *PDDL-temporal* approach that models durative actions. We also analyse the pros and cons on the knowledge engineering procedures used in each approach.

Hierarchical Domain Compilation

This hierarchical domain compilation is based on an extended version of PDDL for handling temporal knowledge in HTN Planners described in (Castillo *et al.* 2006).

When using a hierarchical approach there are two main structures to take into account, tasks and durative actions. Their characteristics, according to the compilation of an e-learning domain, are the following:

To define tasks, the main subject (also called the main task of the course) is formed by ordered subsets of subtasks that have an *IsPartOf* relation with the main one. These subtasks contain others and so on until the subtasks are related to learning activities which are represented as primitive durative actions.

Each task has one or more methods that contain ordered tasks and/or durative actions. Their order is given either by the *IsBasedOn* relation or by additional preconditions based on order rules according to the Honey-Alonso learning style (HoneyAlonso 2002) of each student and its relation with the resource types of the learning activities in the method. For example, if a task is formed by three durative actions, not ordered by *IsBasedOn* relation, and its resource types are *exercise, narrativeText* and *simulation*, then we have two methods with the next preconditions:

```
If the student's learning style is Pragmatic then
  the sequence order is: exercise, simulation and narrativeText
If the student's learning style is Theoretical then
  the sequence order is: narrativeText, simulation and exercise,
```

where *Pragmatic* and *Theoretical* are constants related to the kinds of learning styles for a student according to the Honey-Alonso theory.

In the next lines, we describe the compilation of the *Algorithms* subject (with identifier item-TASK3 in Figure 2). It is related by *IsPartOf* relation with a subject identified as item-TASK5 that *Requires* the durative action item-action6 which is part of Algorithms too. As they are completely related through a *Requires* relation, then item-TASK3 has a unique method with no preconditions and tasks in the explicit order mentioned later.

```
(:task item_TASK3
  :parameters(?id - stId)
 (:method unique
  :precondition()
  :tasks(
    (item_action6 ?id)
    (item_TASK5 ?id))))
```

```
(:task Optional_item_action12
  :parameters(?id - stId)
  (:method yes
   :precondition(availability ?id much)
   :tasks((item_action12 ?id)))
  (:method no
   :precondition()
   :tasks()))
```

On the other hand, and taking into account that this paradigm is based on temporal deadlines, if any of the actions aggregated in the task is related to it through the *References* relation, then an auxiliar task with two methods is created. As in `Optional_item_action12`, the first method does not contain any action or subtask and has no preconditions. The second method must 'invoke' the referenced action only if the student has enough time according to his/her profile.

Finally, to define durative actions we consider that each of them has also conditions related to the student's profile, e.g. a required language level, a high performance, or multimedia availability. Usually, these conditions are assigned to actions with soft preconditions or with the same name but different preconditions and durations. They help adapt a sequence according to the deadlines for each student, which are imposed to each action related to the goal of the course in the problem definition.

PDDL-Conditional Domain Compilation

This approach assumes each learning activity has an utility value that depends on two factors: the student's profile and the learning source type of the activity. According to pedagogical theories, each learning source type is related to the Felder's learning styles, represented in the student's profile (Baldiris *et al.* 2008). So, for each pair <student learning style,activity source type> there is a corresponding utility. We represent learning styles with predicates and the activity utility with a fluent, as it is explained next. The model also assumes that there is a learning object named `fictitious-finish-`*course-name* that contains the tasks required to fulfil the entire course.

We use one predicate for each Felder's learning style. For example, *(sequential ?s - student ?p - profile_level_type)*. The *profile_level_type* can take the value *strong, moderate* or *balanced*. If the system determines that the student is, for

example, *strong sensitive* and *strong active* we would add in the initial state of the PDDL problem the propositions *(active student1 strong)* and *(sensitive student1 strong)*.

Each learning object is translated into a PDDL action in the following way [3]:

- The XML label <title> is used as the action name.
- We define a predicate with the same action name, but adjoining the prefix *task_* and the suffix *_done*. It is added to the action effects and represents the fact that the student has performed such activity and prevents him/her from repeating it.
- The XML label <typicallearningtime> represents the activity duration. We use a fluent to represent the time, *(total_time_student ?s)*, that is increased in the amount of this label in the action effects.
- The XML label <learningsourcetype> represents the activity source type. Its possible values are *lecture, narrativetext, slide, table, index, diagram, figure, graph, exercise, simulation, experiment, questionnaire, problemstatement, selfassessment* and *exam*. We have used the fluent *(reward_student ?s)* to represent the activity utility. Given that it depends on both the student's learning style and the activity source type, we use conditional effects. For example, when the learning source of an activity is a *lecture*, Felder's pedagogical theory says that it is *very good* for *reflective, intuitive* and *verbal* students. So we add the following conditional effects to the PDDL action:

```
(when (reflective ?s strong)
      (increase (reward_student ?s) 40))
(when (intuitive ?s strong)
      (increase (reward_student ?s) 40))
(when (verbal ?s strong)
      (increase (reward_student ?s) 40))
```

  To compute the increasing values of the *reward_student* fluent, we base on a table defined in (Baldiris *et al.* 2008), where rows represent learning source types, columns are the Felder's learning styles, and intersections can take the values: *very good*, *good* or *indifferent*, depending on how the source type adapts to the Felder's style. And, we have converted them into numbers, by some kind of normalization.
- The XML label <relation> defines a relation between two learning activities. We use two of the four types of causal relations defined in the IMS-MD: *Requires* and *IsBasedOn*, with the meaning defined above. In fact, a learning object with an *IsBasedOn* relation is considered as a fictitious action, because the student has to perform only one of the actions in the *or*-condition and both the reward and the total_time remain the same.

Figures 3 and 4 show PDDL actions translated from learning objects with relations of type *Requires* and *IsBasedOn* respectively. The first action describes the activity *simulates-strips-problem*. It requires that the student has already performed activity *reads-classical-planning*, it takes 30 minutes, and it adds the corresponding rewards. The learning source type is *problem* that is *very good* for strong *active, sensitive* and *visual* students and *good* for strong *global* students. We add the precondition `(not (task_strips_done ?s))` to avoid including twice the same action in the plan. The second action represents that a student could perform the activity *simulates-strips-problem* or *experiments-strips-problem* to accomplish the task *task_strips_done*.

```
(:action simulates-strips-problem
   :parameters (?s - student)
   :precondition (and (task_reads-classical-planning_done ?s)
                      (not (task_simulates-strips-problem_done ?s)))
   :effect (and (task_simulates-strips-problem_done ?s)
                (increase (reward_student ?s) 5)
                (increase (total_time_student ?s) 30)
                (when (active ?s strong)
                      (increase (reward_student ?s) 30))
                (when (sensitive ?s strong)
                      (increase (reward_student ?s) 30))
                (when (global ?s strong)
                      (increase (reward_student ?s) 15))
                (when (visual ?s strong)
                      (increase (reward_student ?s) 30))))
```

Figure 3: Example of a PDDL action translated from a learning object with a *Requires* relation.

```
(:action OR-fictitious-strips
   :parameters (?s - student)
   :precondition (and (not (task_strips_done ?s))
                      (or (task_simulates-strips-problem_done ?s)
                          (task_experiments-strips-problem_done ?s)))
   :effect (and (task_strips_done ?s)))
```

Figure 4: Example of a PDDL action translated from a learning object with a *IsBasedOn* relation.

As we said before, the `fictitious-finish-`*course-name* learning object contains, as a *Requires* relations, the tasks required to fulfill the course. This learning object is translated into a fictitious PDDL action with one effect, *(task_course-name_done ?s)*, and its preconditions are the tasks required to complete the course, plus the predicate *(< (total_time_student ?s) (time_threshold_student ?s))*, to avoid the plan to exceed the time limit. This threshold is defined in the planning problem and represents the total time the student can devote to the course. The planning problem has only the goal *(task_course-name_done ?s)*.

This representation allows that any planner that supports full ADL extension (including conditional effects) and fluents can find a solution.

PDDL-Temporal Domain Compilation

This approach follows the same thread presented in the previous conditional compilation *w.r.t.* a non-hierarchical representation of actions in PDDL. However, there are some differences *w.r.t.* the action model:

- As conditional effects are not supported by all existing planners, we do not generate actions with unbound parameters, but fully grounded actions. That is, actions where all the parameters have been instantiated. This means that the name of each predicate needs to include now information about the student. All this process is done automatically. Although this entails rather larger

[3] This compilation takes as input an IMS-MD *Metadata* set.

domains when dealing with many students (only one operator for all the students *vs.* as many grounded actions as students), it simplifies the generation of both preconditions and effects that depend on the student's profile. Now, we do not need preconditions like `(active ?s strong)` nor conditional effects because the action (with all its effects) is generated only if the student is *strong* in the *active* dimension of the learning style. In other words, through a previous automated grounding process, the planning domain will be formed only by those actions that are actually applicable for each student.

- All predicates are generalised to numeric fluents, i.e. all the variable information in the domain is encoded as functions. This means that, instead of using a STRIPS model of actions where preconditions and effects are bi-valued (true/false) predicates, now we can deal with a broader domain of values that allows to keep different levels of knowledge. This increases the expressivity of the model, by allowing us not only metric rewards (e.g. `(increase (reward_Student1) 30)`, like in the conditional compilation) but also having preconditions such as `(>= (Task_Reads-classical-planning_Student1) 50)`. This represents better the fact of: i) achieving marks after executing the tasks, and ii) requiring successful scores before executing tasks.

- This model encodes durations as defined in PDDL2.1, and its successors, by using the *:duration*. This value is directly taken from the *typical learning time* metadata of the learning object. Thus, the PDDL domain can be subsequently used by any temporal planner. Nevertheless, this compilation also has the ability to model time as in the conditional compilation; that is, by means of an artificial fluent `(total_time)` that represents the time-line. The advantage of doing this is that this approach provides a domain compilation valid for existing temporal and non-temporal metric planners.

- The hierarchical structure is flatly encoded by means of two dummy actions, *Start* and *End*, that represent the aggregation activity. *Start* contains the preconditions of the aggregation activity and *End* its effects. On the other hand, the actions generated for all the aggregated objects have that *Start* as precondition. Obviously, both *Start* and *End* have duration 0. Recalling the example depicted in Figure 2, Figure 5 shows an example of the three actions that are generated when encoding the hierarchical relations in a flat structure.

- The utilisation of numeric fluents in actions makes the inclusion of metric resources and their cost easier, as traditionally used in P&S. Particularly, this approach can also model the cost of each action, in terms of the resources used, by simply adding a new effect such as `(increase (resource_cost_Computer) value)`. The inclusion of the resources cost will later allow the user to define more flexible metrics to be optimised in the planning problem.

```
(:durative-action Start_ITEM-task3_Std1 ;; Algorithms
   :parameters ()
   :duration (= ?duration 0)
   :condition (and (at start (= (Start_ORG-s2ctest2_Std1_done) 1))
                   (at start (= (Start_ITEM-task5_Std1_done) 1))
                   (at start (= (Start_ITEM-task3_Std1_done) 0)))
   :effect (and (at end (increase (Start_ITEM-task3_Std1_done) 1))))

(:durative-action End_ITEM-task3_Std1 ;; Algorithms
   :parameters ()
   :duration (= ?duration 0)
   :condition (and (at start (= (Start_ITEM-task3_Std1_done) 1))
                   (at start (= (Start_ITEM-task6_Std1_done) 1))
                   (at start (= (End_ITEM-task3_Std1_done) 0)))
   :effect (and (at end (increase (End_ITEM-task3_Std1_done) 1))
                increase other numeric expressions or resource_costs))

(:durative-action ITEM-action6_Std1 ;; Basic Algorithms Lecture
   :parameters ()
   :duration (= ?duration 9)
   :condition (and (at start (= (Start_ITEM-task3_Std1_done) 1))
                   (at start (= (ITEM-action6_Std1_done) 0)))
   :effect (and (at end (increase (ITEM-action6_Std1_done) 1))
                increase other numeric expressions or resource_costs))
```

Figure 5: Durative actions generated for the learning objects of Figure 2.

Problems Compilation

Once the domain is generated, we need to define problems in such a way that, when the planner solves them, each plan represents a learning design for a particular student. That is, the sequence of learning actions a student should perform in order to complete the course. Usually, LMS have mechanisms based on standards to access the relevant student information for the designs. IMS-LIP has become a standard for storing such student information. But, again, this standard is too generic and tries to cover too many aspects. Therefore, it is necessary to select the relevant student's characteristics required for our planning problems and the XML fields that contain them.

This section describes a proposal of IMS-LIP schema for translating it into a planning problem. The proposal is valid for the hierarchical domain and the PDDL domain with fluents and conditional effects, although the translated propositions, obviously, differ in each domain in relation to the domain predicates. So far, the planning problems for the temporal domain must be defined separately because the grounded domain makes an automatic translation difficult. After all, when working with grounded actions, both the planning domain and problem are packed together as the grounded actions in the domain are only valid for that particular planning problem.

On the one hand, planning problems include propositions to represent the objects, the initial state, the goals and a metric to optimise. In our domains, the objects represent the student's information for the learning design. The initial state represents the student's profile, the initial values of the fluents, the previous knowledge of the student, the language of the course, and some other information (e.g. performance, equipment, availability, etc.). The goal is usually to pass the entire course or a part of it.

On the other hand, the core structures of the IMS-LIP are based upon: accessibility information, activities, affiliations, competencies, goals, identifications, interests, qualifications, certifications and licences, relationship, security

keys, and transcripts. Within each category several data elements and structures are defined. Some of these are specified explicitly as data types (language strings, for the most part) and others are defined as recursive hierarchical structures. Thus, the question is how to match both structures, the IMS-LIP and the planning problem ones, so that, an automatic translation compiles an IMS-LIP file into a planning problem. Table 1 shows the XML fields we have used to allow this compilation. The first column represents the IMS-LIP code and the second column represents the corresponding translation into the planning problem.

IMS-LIP	Planning Problem
<identification><name> <contentype><referential> <indexid>student1	*:objects* student1
<accessibility> <preference> <typename><typevalue> Learner_Style_Processing <prefcode>reflective.strong	*:inits* (reflective student1 strong)
<accessibility><preference> <typename><typevalue> Learner_Style_HoneyAlonso <prefcode>theoretical.strong	*:inits* (honeyAlonso student1 theoretical)
<goal><typename> <tyvalue>AI-course <contentype><temporal> <typename>Time_Threshold <temporalfield>3881	*:inits* (= (time_threshold student1) 3811) *:goals* (task_AI-course_done student1)
<activity><typename> <tyvalue>Task <learningactivityref> <text>graph_theory	*:inits* (task_graph_theory_done student1)
<accessibility><language> <typename><tyvalue>English	*:inits* (language_level English student1 high)
<competency><contentype> <referential><indexid> performanceLevel <description> <short>High	*:inits* (performance_level student1 high)

Table 1: Example of a problem compilation from an IMS-LIP. Irrelevant information has been eliminated.

Approaches Comparison

Table 2 shows the differences between the three approaches regarding some characteristics. The first rows represent where or how each planning feature is defined. For example, the *Hierarchical* domain is translated from a course defined in Moodle format and the *PDDL-Temporal* one can be defined either in Moodle or in IMS-MD format. The translation is fully automated in all cases. 'Tool' means that the characteristic is defined through the Tool we have implemented. The planning goals in the *Hierarchical* approaches are defined in the IMS-LIP, while in the *PDDL-Conditional* one are defined in a learning object of the Metadata set. The row *Deadline definition* represents where the time limit each student can devote to the course is defined. For example, in the *Hierarchical* and *PDDL-Conditional* approaches, it is a field in the IMS-LIP that is automatically translated. The row *Prerequisite definition* refers to the previous knowledge the student should have in order to follow the course. The row *LOM relations* means the types of relations, according to LOM terminology, supported for the approaches. *Soft preconditions* refers to the fact that the learning design can contain activities that, without being mandatory, provide some benefit to the student. This is possible through the methods in the *Hierarchical* approaches and through the precondition *(< (total_time_student ?s) (time_threshold_student ?s))* in the *PDDL-Conditional* representation. *Time management* represents how the approaches deal with time. The *Hierarchical* and *PDDL-Temporal* approaches use durative actions while the *PDDL-Conditional* uses fluents. The *PDDL-Temporal* can also compile the domain using a fluent to represent time instead of durative actions. The row *Metrics* means whether the approach can manage quality metrics or not. The last row represents the planner required to solve the problems modelled by the approach. SIADEX (Castillo *et al.* 2006) is the only planner able to generate learning designs in the *Hierarchical* approach, because of the input language. So far, there is no standard language for representing hierarchical domains in PDDL. The other two approaches compile the domains into PDDL, so any planner that supports fluents, metrics, and conditional effects, in the case of *PDDL-Conditional*, or durative actions in the case of *PDDL-Temporal*, can solve the problems.

Characteristic	Hierarchical	PDDL-Conditional	PDDL-Temporal
Domain definition	Moodle	IMS-MD	Both
Problem definition	Moodle and IMS-LIP	IMS-LIP	Tool
Goal definition	IMS-LIP	LO in MD	Problem (Tool)
Deadline definition	IMS-LIP	IMS-LIP	Problem (Tool)
Prerequisite definition	IMS-LIP	IMS-LIP	Problem (Tool)
LOM relations	All	IsBasedOn Requires	All
Students' profile	Honey-Alonso	Felder	Both
Soft preconditions	Method	Domain	-
Time management	Durative actions	Fluent	Both
Metric	No	Yes	Yes
Planner	SIADEX	Conditional effect Metrics	Temporal Metrics

Table 2: Approaches comparison.

The *Hierarchical* approach permits modelling more learning features, including durative actions, but only the planner SIADEX can solve the problems. Also, the methods have to be manually defined. Also, it cannot deal with quality metrics. The other approaches use PDDL and can deal with metrics such as minimizing the total time the student devotes to the course. However, current state-of-the-art planners cannot manage maximizing metrics, so a metric for maximizing the total utility that the learning activities report to the student is not easily applicable. The *PDDL-Temporal* approach automatically generates grounded domains avoiding the use of conditional effects, but it makes ulterior domain modifications difficult, as, for example, trying to find ways for maximizing the utility.

Tool

Once the three translation modules have been presented, we describe the tool that supports the user on generating the planning files. As indicated in Figure 1, the tool comprises two parts and its main goals are twofold. First, the tool acts as an interface for the translators, thus making this process simple and transparent to the user. Second, the tool provides a graphic visualization of the learning objects and their relations, and also allows the designer to modify and tune them by means of intuitive drag&drop graphic components and user-friendly input forms.

Support Interface for Translation

The tool contains options for both importing and exporting files in standard e-learning formats, together with translation support to planning files. Particularly, we can easily import/export the learning objects encoded as IMS-MD in dotLRN or in Moodle XML files. As an example, Figure 6 shows a snapshot of the tool when importing a simple Moodle course with the objects depicted in Figure 2. The possibility of importing learning objects from common standards is very convenient as it allows the designer to reuse many of the objects available in web repositories. After that, the tool uses the three different compilation methods described in the previous section to generate the planning domains and problems accordingly.

Graphic Visualization and *Tuning* of the Learning Objects

The second part of the tool focuses on modelling e-learning courses, acting as a complementary module to specify and facilitate the completion and extension of metadata records of learning objects, specially those related to the structural and logical relationships that are essential for planning. Loosely speaking, the tool offers a much more intuitive representation of the learning objects by using graphic elements (see Figure 6) rather than the XML files (see Figure 2). This is interesting as we can see at a glance the hierarchical structure, the aggregated objects, the students' profiles, their relationships and also helps notice some inconsistencies, such as circular dependencies between learning objects (e.g. A and B Require each other).

A clear advantage of our tool is that it can be used to improve the quality of the learning objects, at least from the planning perspective. The e-learning standards include much information within the objects, in the form of metadata, but they are not always directly usable in planning. Actually, some of the items are not concerned with planning, like the keywords, the format or the source of the objects. Others are not equivalent to the same named items in planning, such as the resources: a resource in e-learning may be a URL the student needs to visit, but not a shared resource that imposes additional constraints and costs to the plan. Consequently, we can use the tool to complete and tune the metadata labelling of the learning objects, making them more accurate *w.r.t.* i) information about the student's profile, ii) required resources, iii) typical learning time, i.e. duration, and iv) relationships among objects and their types. After all, the more accurate the metadata of the objects is, the better for the planner —it will have more opportunities to find a plan better adapted to the student. Figure 7 shows one of the forms that allow to input basic information about the learning object (from the planning point of view), min and max duration, requirements on profiles and previous concepts, necessary resources, etc. With all this information, and once the students' information is modelled, the tool performs a temporal domain compilation and generates both the domain and the problem files in PDDL format, ready to use by any existing domain-independent planner.

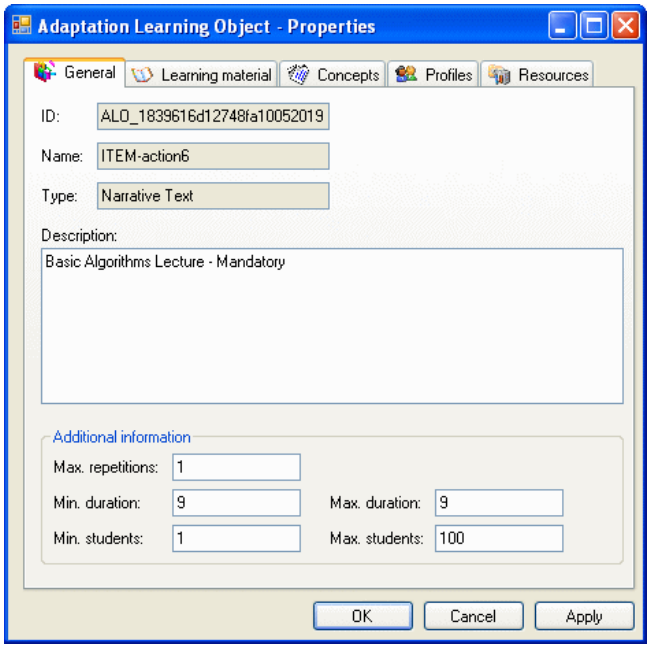

Figure 7: Input form with information about the learning object 'Basic Algorithms Lecture'.

Use of Plans

Each plan generated by a domain-independent planner, using a domain and a problem as described above, represents the learning design that best suits the student whose characteristics are modeled in the problem. Learning platforms include tools for executing the designs when the design is represented in a specific language. For example, dotLRN interprets IMS-LD, whereas Moodle has its own templates. We have implemented two more translators: from the plans of non-hierarchical planners to IMS-LD and from the plans of a hierarchical planner to the Moodle templates.

The first translator compiles a plan into an IMS-LD that dotLRN is able to execute. This is a zip file that contains the input resources (learning objects), as well as the learning design (output of the planner). The first part is basically a copy of the input resources that is usually contained in a directory. The second part consists of an XML file. Next, we describe the main fields of this file (some others are easily filled in from this information) and how they are generated from the domain, problem and plan:

- Objectives: these are filled with the name of the goals of the problem. The problem goals are always the main effects of the final action of any domain.
- Prerequisites: these are the initial conditions on previous knowledge that is required to follow this course. They are the links to learning objects of other courses, or objectives of other courses. This will allow us to perform multiple course planning in the future.
- Roles: in this case, the only role is that of the learner, the student for whom the learning design is generated.

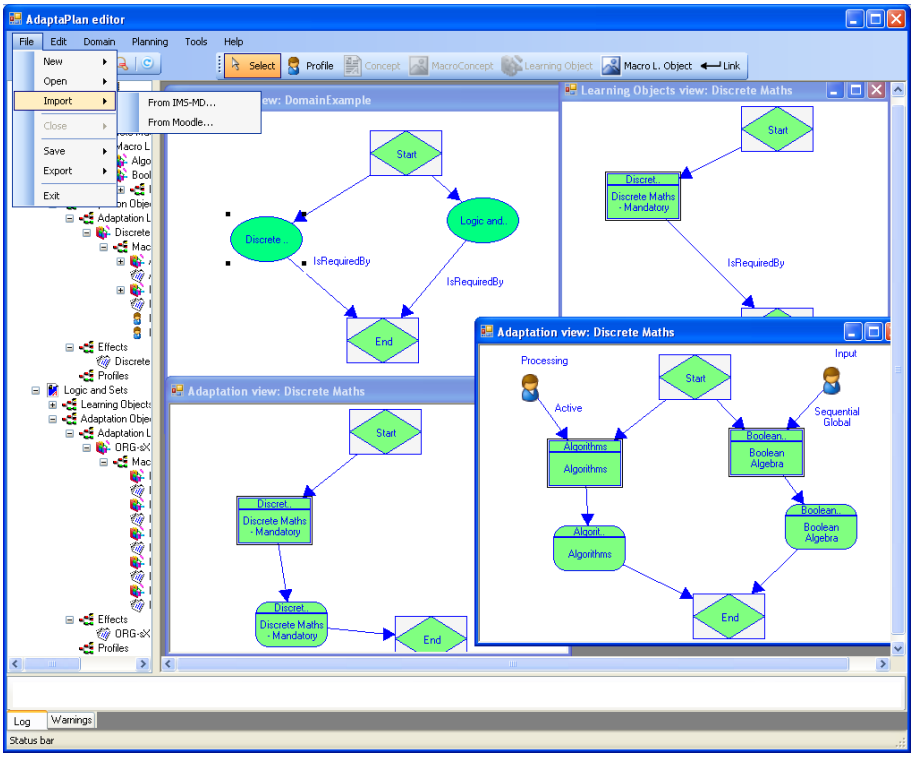

Figure 6: Snapshot of the tool.

- Activities: for each action in the plan, an IMS-LD activity entry is generated. This field is just an enumeration of those actions, and each one also includes a link to the corresponding learning object. Fictitious actions are omitted.
- Activity-structure: this IMS-LD concept relates to the plan itself. So, here the sequence of actions in the plan is represented as such. Given that the standard allows other control structures, such as conditional plans with branches, in the future we will study how to generate conditional plans and the effect it has on the fact that students follow different alternatives.
- Resources: for each learning object in the input IMS-MD, a resource, that can be or not used by the plan, is defined.

Once the translator generates the zip file, it can be uploaded to dotLRN and be used by any student after a sequence of bookkeeping activities: defining the student's preferences, defining relevant roles of the course, initiating the execution of the course and so on.

The second translator compiles the hierarchical plan into a Moodle template. This template describes an XML file which is automatically related with a course by Moodle. The XML document contains several items with a student's identifier and a related action, as in next lines.

```
<item>
  <studentId> student1 </studentId>
  <actionId> ITEM-action1 <actionId>
</item>
<item>
  <studentId> student1 </studentId>
  <actionId> ITEM-action6 <actionId>
</item>
```

The actions related with each student's identifier are not subjects of a course, but learning activities, i.e. durative actions which were previously stored in Moodle database using IMS-MD standard. The order in which items appear in the document correspond to the plan with the learning activities sequencing obtained for each student. Internally, hierarchy of subjects is provided by Moodle according to the information previously stored.

Moodle permits us to deal with collaborative plans. If items of several students are interspersed, then dependency between actions of a previous student and the next one must be taken into account by the platform. Obtaining collaborative plans to take advantage of this characteristic is a task to be done in a short future.

Conclusions and Future Work

E-learning is about designing a sequence of learning activities a student needs to perform in order to complete a course. Principally, this involves three main issues: course definition, student's learning information and learning design execution. There are languages to represent all of them based on XML schemata, but an important effort needs to be done to be fully automated.

This paper has proposed a three-approach procedure to interpret and translate e-learning tasks into automated planning. A course definition is represented as a planning domain, the student's learning information as a planning problem for that domain and the learning design as the plan generated by a domain-independent planner when solving that

problem. We have implemented translators from the corresponding e-learning languages into three different kinds of planning domains and problems. These three domain reasoners allow different planners to automatically generate valid learning designs in a few seconds. However, we have detected three main drawbacks. First, e-learning languages are too generic and try to cover too many aspects, making the implementation of general and suitable translators for all LMS very difficult. Second, in spite of the expressive power of e-learning languages, there are still few aspects that cannot be represented and are essential in P&S. For example, the definition of the resources involved in tasks, their costs, the temporal constraints on availability and how these resources are to be managed are important lacks in e-learning languages. Finally, our domain modelling allows us to generate valid learning designs, but they cannot guarantee optimal plans in terms of utility to the student.

In the future we want to find solutions to overcome the previous drawbacks by addressing two parallel lines. Firstly, we are interested in coming up with more expressive models of actions for planning e-learning activities. This will increase the opportunities to: i) deal with course composition, and ii) validate and resolve courses with similar but incommensurate learning objects. Secondly, we want to extend the tool to assist the course designer in making sure that the same naming conventions are used. As one of the anonymous reviewers suggested, the adoption of a common ontology can be very useful (Kontopoulos *et al.* 2008).

References

Baldiris, S.; Santos, O.; Barrera, C.; J.G., J. B.; Velez, J.; and 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 (IJCSA)*.

Brusilovsky, P., and Vassileva, J. 2003. Course Sequencing Techinques for Large-Scale Web-Based Education. *International Journal Continuing Engeenering Education and Lifelong Learning* 13(1/2):75–94.

Camacho, D.; R-Moreno, M.; and Obieta, U. 2007. CAMOU: A Simple Integrated e-Learning and Planning Techniques Tool.

Castillo, L.; Fernández-Olivares, J.; García-Pérez, O.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006*, 63–72.

Castillo, L.; Morales, L.; Gonzalez-Ferrer, A.; Fdez-Olivares, J.; Borrajo, D.; and Onaindia, E. 2009. Automatic generation of temporal planning domains for e-learning problems. *Journal of Scheduling*. Accepted.

Felder, R. M. 1996. Matters of style. *ASEE Prism* 6(4):18–23.

HoneyAlonso. 2002. Honey alonso learning style theoretical basis in spanish. Available at http://www.estilosdeaprendizaje.es/menuprinc2.htm.

IMS-GLC. 2001-2009. Ims specifications. Available at http://www.imsglobal.org.

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

LOM. 2002. Draft standard for learning object metadata. IEEE. 15 july 2002. 6 oct. 2007. Available at http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf.

Mohan, P.; Greer, J.; and McCalla, G. 2003. Instructional Planning with Learning Objects.

Peachy, D., and McCalla, G. 1986. Using Planning Techniques in Intelligent Tutoring Systems. *International Journal of Man-Machine Studies* 24(1):77–98.

Ullrich, C., and Melis, E. 2009. Pedagogically Founded Courseware Generation Based on HTN-planning. *Expert Systems with Applications* 36(5):9319–9332.

Vrakas, D.; Tsoumakas, G.; Kokkoras, F.; Bassiliades, N.; Vlahavas, I.; and Anagnostopoulos, D. 2007. PASER: a curricula synthesis system based on automated problem solving. *Int. Journal on Teaching and Case Studies, Special Issue on "Information Systems: the New Research Agenda, the Emerging Curriculum and the New Teaching Paradigm"* 1(1/2):159–170.

Wisher, R. 2009. *Sharable Content Object Reference Model(SCORM) 2004 4th Edition Documentation Suite*. ADL.