# Knowledge engineering and planning for the automated synthesis of customized learning designs

Luis Castillo[1], Lluvia Morales[1], Arturo González-Ferrer[2], and Juan Fdez-Olivares[1] and Óscar García-Pérez[3]
[1]Dpto. Ciencias de la Computación e I.A.University of Granada
[2]Centro de Enseñanzas Virtuales. University of Granada
[3]IActive Intelligent Solutions

**Abstract.** This paper describes an approach to automatically obtain an HTN planning domain from a well structured learning objects repository and also to apply an HTN planner to obtain IMS Learning Designs adapted to the features and needs of every student.

## 1 Introduction

Nowadays, distance learning is positioning as a key tool not only for graduate courses but also for professionals continuing education. In these areas, the heterogeneity of students, their different performance and needs and previous studies force current e-learning platforms to highlight the issue of customizing learning designs so that every student may optimally exploit the contents of a given course. This is not new, and the need to adapt learning designs is carefully described in current standards for learning management systems (LMS). Educational metadata (IEEE-LOM [1] or IMS-MD [5]) allows instructors to classify learning resources according to a set of variables. Student profiles (IMS-LIP [5]) are also represented to gather information about their features.And, finally, learning designs (IMS-LD [5]) allow instructors to adapt the learning path and the use of learning resources to the features and capabilities of every student.

The use of these standards, amongst others, is fostering a common language and new interoperability capabilities between all the entities involved in e-learning activities. Even more, different learning objects could be potentially shared between different platforms. However, the process of building a learning design is a very complex task, that must be manually developed by the instructor.

This paper focuses on making the life of instructors easier, thanks to the use of artificial intelligence planning techniques [4, 3] able to automatically obtain a learning design customized to every student needs and features. These techniques have been usually employed to help experts of different fields to define their strategic plans like in aerospatial domains , civil emergencies or military campaigns . However, they are also specially useful for the design of learning paths, since they can both explore all the possibilities of the available learning resources, its different learning objects and their features, and also take into account the features and needs of every student in order to elaborate, like in those strategic plans, the optimal learning path for every student.

We must say that the use of artificial intelligence planning techniques in real applications imply a great knowledge engineering effort in order to acquire and validate the available know-how of every domain and to encode this knowledge into a set of rules or protocols which is usually named the planning domain. A planning domain is the core of any planning application that guides the search effort of the planner and it is usually written in the Planning Domain Description Language [6] or any of its flavours [3]. This important effort has been an obstacle for the practical application of planning techniques and the technical part of this paper is devoted to show that this planning domain may be automatically generated from a well structured domain like the learning objects repository of a LMS. In order to do that, we propose an exhaustive labeling of learning objects making use of the IMS-MD or IEEE-LOM standards and an inference procedure that explores all the metadata and relations to generate a valid planning domain. Later on, a state-of-the art planning algorithm might be used to obtain a customized learning design for every student. Given that most LMS are intended to use these standard metadata, our approach could be directly used in any of them just by checking for a correct labeling of learning objects. These contributions are aligned towards an important horizon: enable end users (instructors and students) to easily adopt e-learning standards at a low cost.

## 2 HTN planning foundations

In order to better understand the main contributions of this paper, a brief introduction to HTN planning techniques [4] is presented first. HTN (Hierarchical Task Network) planning is a family of artificial intelligence planners that have shown to be very powerful in practical applications on very different domains. HTN planning paradigm is based on the same three concepts than any other planning approach. The initial state is a set of literals that describe the facts that are true at the beginning of the problem; this would be the students' profile. The goal is a description of what we want to achieve with a plan, that is, the learning goals.The domain is the set of available actions or rules to achieve the goals. In our case, the available learning objects.

### 2.1 HTN planning domains

HTN planning domains are designed in terms of a hierarchy of compositional activities. Lowest level activities, named actions or primitive operators, are non-decomposable activities which basically encode changes in the environment of the problem. In our approach, these primitive operators are represented as PDDL 2.1 level 3 durative actions. On the other hand, high level activities, named tasks, are compound actions that may be decomposed into lower level activities. Depending on the problem at hand, every task may be decomposed following different schemes, or methods, into different sets of sub-activities. These sub-activities may be either tasks, which could be further decomposed, or just actions. Tasks and their, possibly multiple, decompositions encode domain dependent rules for obtaining a plan, that can only be composed of primitive actions. Unlike non HTN planners, HTN goals are not specified as a well formed formula that must be made true by the plan from the initial state. Instead, goals are described as a partially ordered set of tasks that need to be carried out. And finally, the main HTN planning algorithm takes the set of tasks to be achieved, explores the space of possible decompositions replacing a given task by its component activities, until the set of goal tasks is transformed into a set of only primitive actions that make up the plan.

HTN planning domains are usually written after a knowledge engineering stage in which the know-how of the problem is studied and formally represented [10, 3]. This stage requires a strong commitment of domain's experts and a deep knowledge of planning techniques, so it is not an easy task. However, in problems with an underlying strongly structured knowledge like a learning objects repository with an exhaustive metadata labeling, this domain could be automatically extracted as will be shown in the following sections.

## 3 Our approach

The main idea behind this approach is that AI planning techniques may be used to automatically generate a customized learning design based on the following assumptions: (1) The learning objects repository is labeled using a extensive set of standard metadata that is described along this section. (2) The instructor explores the repository and define the learning objectives of a given course. (3) Our system explore the different databases of users profiles, learning objects and learning objectives and generate the necessary PDDL files [6, 3] for our HTN planner to run. The planner is executed and a customized learning plan is obtained for every student registered at the same course. (4) The learning plan is translated into a form playable or understandable by the LMS. (5) The plan is executed (or played) by the student to follow the course adapted to its own features and needs.

In order to guarantee a valid extraction of an HTN planning domain and a successful adaptation of the learning path, at least the following set of metadata are required to be present in the labeling of the learning objects[1]:

**Hierarchical structure.** Hierarchical relations of the form chapter/sub-chapter/lesson, being lesson the atomic part of the hierarchy, are encoded by means of the `is-part-of` relational metadata. This allows to encode hierarchical dependencies between learning objects. A learning path, that is, the sequence

[1] There is a wider variety of metadata that allow for a greater adaptation capability to the features of every student. They have not been included due to lack of space

of learning objects that is to be followed by student will only be made up of atomic objects. This means that compound objects might have no content, except those included in their constituent atomic objects and they primarily act as the underlying structure of the course. (Figure 1).

**Ordering relations.** The order relation or sequence between learning objects defined by the instructor, in the case that they exist, are encoded by means of the `is-based-on` relational metadata. Figure 1 shows a simple example so far.

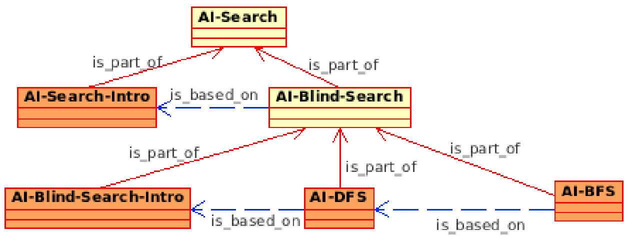

**Fig. 1.** A simple labeling of learning objects showing a piece of a classic Artificial Intelligence course chapter devoted to search: After a brief introduction, the sections about depth-first search (DFS) and breadth-first search (BFS) exactly in this order. Lowest level objects (atomic) appear shadowed

**Content dependencies.** Sometimes, the content of a given chapter or sub-chapter depends on other chapters of the same repository. Since the student may or may not have background knowledge on these dependencies, they are encoded my means of the relational metadata `requires`. For example, the chapter `AI-Search` depends on knowledge about graphs, that belongs to a learning object of another course. This dependency is encoded to allow the planning algorithm to reason about the convenience or not of including a chapter about graphs in a given learning path: if the student does not know about graphs, it would be strongly required to pass first this chapter, otherwise, it would be ignored. (Figure 3)

**Optional lessons.** These are lessons that may be included or not in a learning path depending on some conditions, usually the global time span of the course. This is encoded by means of the general metadata `coverage` that is labeled with the constant `optional`. If this metadata is empty, then the learning object is intended to be mandatory.

**Different languages.** Our approach is also intended to cope with repositories handling different languages so that the planner may or not may select some learning objects depending on a student's knowledge of other languages. It is encoded with the general metadata `language`.

**Typical Learning Time.** The educational metadata is a very important issue to successfully encode a learning path given the temporal constraints imposed by the course, the student or both.

**Type of resources.** Every resource, that is, a learning object, in the repository must be labeled with the educational metadata `learning-resource-type` (a lesson, an example, an excercise, etc).

**Hardware/Software requirements.** In the case that a given learning object would require special hardware or software features (like multimedia files, for example) this could be used for the planner to reason about its inclusion or not in the learning path depending on the declared HW/SW platform of every student. This is encoded in the technical metadata `other-platform-requirements`.

These are standard IEEE-LOM [1] metadata and they are needed to ensure a correct domain extraction from the repository, so it is not a heavy requirement of our approach, since they are supposed to be present in most standard learning objects repositories.

## 3.1 Domain extraction in a simple case

In the simplest case (Figure 1), a repository containing just hierarchical and sequencing metadata may be intuitively translated into an HTN domain as those presented earlier just by exploring these relations. Compound objects (those with any "child object") would be translated into a compound task and simple objects (those with no children) would be translated into non-decomposable actions. Therefore, the simple repository shown in Figure 1 would be translated into the HTN domain shown in Figure 2

```
(:task AI-Search
 :parameters (?student)
 (:method One
  :tasks (
         (AI-Search-Intro ?student)
         (AI-Blind-Search ?student))))
(:task AI-Blind-Search
 :parameters (?student)
 (:method One
  :tasks (
         (AI-Blind-Search-Intro ?student)
         (AI-DFS ?student)
         (AI-BFS ?student))))
```

```
(:durative-action AI-Search-Intro
 :parameters(?student)
 :duration (= ?duration
          (typical-learning-time AI-Search-Intro))
 :condition()
 :effect(passed ?student AI-Search-Intro))
(:durative-action AI-DFS
 :parameters(?student)
 :duration (= ?duration
          (typical-learning-time AI-DFS))
 :condition()
 :effect(passed ?student AI-DFS))
```

**Fig. 2.** Part of the domain extracted from the sample repository of Figure 1.

## 3.2 Domain extraction in complex general cases

The simple domain extraction procedure roughly sketched before is too simple and it does not allow for a full adaptation of the learning path, taking into account the full set of metadata present in the repository (Figure 3). This section fully describe a domain extraction procedure that completely fits into the expected adaptation scheme represented in these metadata. Let us consider the repository shown in Figure 3

**Extracting primitive actions.** HTN primitive actions are extracted from those learning objects that have no children. These actions will take into account the following issues. The duration of the action will be its typical learning time. If there are some special hardware requirements or it has been written in a

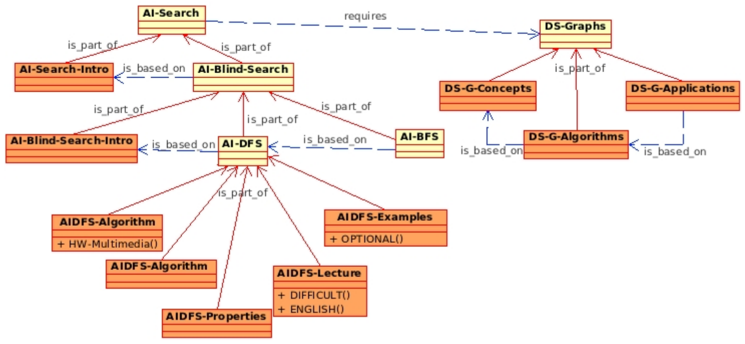

**Fig. 3.** A labeling of learning objects more slightly more complex than Figure 1. Lowest level objects (atomic) appear shadowed.

language different than the common language of the course, then the list of preconditions will include these conditions for the action to be included. Figure 4 shows two actions that exhibit these preconditions so they will only be included in the learning path if the profile of the student meets these conditions.

```
(:durative-action AIDFS-Algorithm
 :parameters(?student)
 :duration (= ?duration
  (typical-learning-time AIDFS-Algorithm))
 :condition(hardware ?student multimedia)
 :effect(passed ?student AIDFS-Algorithm))
```

```
(:durative-action AIDFS-Lecture
 :parameters(?student)
 :duration (= ?duration
           (typical-learning-time AIDFS-Lecture))
 :condition(>= (mark ?student english) 50)
 :effect(passed ?student AIDFS-Lecture))
```

**Fig. 4.** Action `AIDFS-Algorithm` requires the student hardware platform to have multimedia capabilities. Action `AIDFS-lecture` is written in english, a foreign language for the student, and it requires the student to have a satisfactory mark registered in its profile (at least 50 out of 100).

For every atomic learning object labeled as "optional", a new task is created with two different methods, one of them includes its corresponding action and the other does not. For example, Figure 5 shows how the optional object `AIDFS-Examples` is treated.

As may be seen in Figure 3, there may be more than one atomic object with the same name (i.e. there are two objects with name `AIDFS-Algorithm`). This means that they are different ways of performing the same learning act and, probably, under different conditions. This allows the student to follow a given lesson although the lesson offered to each student might be different depending on their context. This is encoded as an additional compound task that includes a unique method containing a single action. There will be a primitive action for each atomic object so that, the compound task forces the introduction of one of these actions that will be found by the planner by search and backtracking in the case that the conditions of the actions are not met (see Figure 6).

**Extracting compound tasks.** The previous domain extraction procedure allows to generate the primitive actions of an HTN domain and some additional

```
(:task OPTIONAL-AIDFS-Examples
 :parameters (?student)
 (:method Yes
  :precondition ()
  :tasks (
          (AIDFS-Examples ?student)))
 (:method No
  :precondition ()
  :tasks ()))
```

```
(:durative-action AIDFS-Examples
 :parameters(?student)
 :duration (= ?duration
           (typical-learning-time AIDFS-Example))
 :condition()
 :effect(passed ?student AIDFS-Example))
```

**Fig. 5.** Action `AIDFS-Examples` is optional. This is encoded as a compound task with two alternative decompositions. The first one, labeled as "Yes" tries to include the object `AIDFS-Examples`. If a backtracking is produced during the search, then the method labeled as "No" introduces an empty decomposition, that is, it does not include the object.

```
(:task MULTIPLE-AIDFS-Algorithm
 :parameters (?student)
 (:method Unique
  :precondition ()
  :tasks (
          (AIDFS-Algorithm ?student))))
```

```
(:durative-action AIDFS-Algorithm
 :parameters(?student)
 :duration (= ?duration
  (typical-learning-time AIDFS-Algorithm))
 :condition(hardware ?student multimedia)
 :effect(passed ?student AIDFS-Algorithm))
(:durative-action AIDFS-Algorithm
 :parameters(?student)
 :duration (= ?duration
  (typical-learning-time AIDFS-Algorithm))
 :condition()
 :effect(passed ?student AIDFS-Algorithm))
```

**Fig. 6.** Part of the domain extracted from the sample repository of Figure 3. Action `AIDFS-Algorithm` may be included in any of the forms present in the domain to adapt the learning path to the existing conditions

compound tasks to encode part of the adaptation scheme. This will enable the planner to adapt a learning path to the individual features of every student. However, there are still more possibilities to encode additional adaptation schemes in the repository so that the search capability of the planner is increased. They are related to the decomposition of a compound task. There may be compound actions, like `DS-Graphs` in Figure 3 whose constituents parts are fully ordered and, therefore, they will always be included in the same order in any learning path. However, other compound tasks like `AI-DFS` do not have its constituents objects fully ordered. This means that the order in which these objects will be included in a learning path is not always the same and it may depend on some external conditions. In order to represent this, implicit ordering relations are defined in our approach to encode different orderings for every compound task based on the `learning-resource-type` of every object. For example, the following rule

```
((TRUE) (Problem-statement Simulation Experiment Exercise Lecture))
```

would mean that in every possible situation in which the order of the component objects is not explicitly defined by the instructor, the ordering in which they appear in the task will be the following: first the object labeled with `learning-resource-type` equal to `problem-statement`, then those labeled as `simulation`, those labeled as `experiment`, the objects labeled as `exercise` and at the end, those labeled as `lecture`.These other rules encode a more interesting example:

```
((Honey-Alonso-Learning-Type ?student Theorist)
   (Problem-statement Simulation Experiment Exercise Lecture))
((Honey-Alonso-Learning-Type ?student Pragmatic)
   (Simulation Experiment ExerciseProblem-statement Lecture))
```

They mean that the decomposition of a compound object depends on the registered Honey-Alonso learning profile of every student. For example, these two previous rules would produce the decomposition scheme for task `AI-DFS` shown in Figure 7.

```
(:task AI-DFS
 :parameters (?student)
 (:method Pragmatic
  :precondition (learning-type ?student pragmatic)
  :tasks (
(MULTIPLE-AIDFS-Algorithm ?student)
        (AIDFS-Examples ?student)
        (AIDFS-Properties ?student)
        (AIDFS-Lecture ?student)))
```

```
 (:method Theorist
  :precondition (learning-type ?student theorist)
  :tasks (
        (AIDFS-Properties ?student)
(MULTIPLE-AIDFS-Algorithm ?student)
        (AIDFS-Examples ?student)
        (AIDFS-Lecture ?student))))
```

**Fig. 7.** Task `AI-DFS` is decomposed depending on the Honey-Alonso learning profile of the student

And finally, there is the case that a compound object requires another object that belong to any other course. Figure 3 shows that the object `AI-Search` depends on the object `DS-Graphs` that belongs to another course, let say Data Structures. In this case, the task AI-Search includes two different decompositions, one of then for the case that the student has successfully passed this required object, and the other one for the case that the student has not passed this object and thus, will have to be included in his/her learning path.

In summary, this section has shown that a valid HTN domain may be extracted from a well structured learning objects repository just by exploring a set of standard metadata present in most LMS[2].

## 4 Obtaining a plan

The application of this procedure to a learning objects repository would produce a file named "domain.pddl" that is one of the components for any PDDL-compliant planner. This file will be the same for every student since it only contains the translation of the learning objects repository into a PDDL domain. However, for the planner to run, there is another file that must be present. It is usually named "problem.pddl" and it encodes, both, the initial state and the goal of the problem. In terms of a LMS, the initial state encodes student profiles and the goal encodes the learning goals.

Students profiles are extracted from the LMS' databases following the IMS-LIP [5] standard or any other equivalent formalism. These profiles will contain all the available information about the student that will make the planner to search and backtrack amongst the available tasks and actions in the translated domain

[2] There is a preamble of the HTN planning domain in PDDL, but it is practically always the same and it is not included here.

and, therefore, to optimally adapt the desired learning path to its features and needs.

Learning goals are defined by the instructor amongst the list of compound tasks available in the domain (i.e. the highest level learning objects) and they will appear totally ordered in the goal section of the pddl problem. Figure 8 shows a piece of this problem. As may be seen, there are two students, Peter and Clark, Peter doest not have any temporal constraint[3] to end the course, but Clark needs to end the course in less than 320 minutes.

```
(define (problem simple)
(:domain test)
(:objects
  Peter, Clark - student)
(:init
  (learning-type Peter pragmatic)
  (= (mark Peter english) 50)
  (passed Peter DS-Graphs)
  (hardware Peter multimedia) ...)
```

```
(:tasks-goal
  :tasks [
    (AI-Search Peter)
    ((<= ?end 320) (AI-Search Clark))
  ]
)
```

**Fig. 8.** The problem of the PDDL scenario is also automatically extracted from the LMS databases, both the initial state (students profiles) and the goal (learning goals asserted by the instructor).

Once the domain and the problem have been translated from the LMS repository and databases into PDDL compliant files, the HTN planner [3] is executed and an adapted learning path is obtained for each of the students included in the problem. This plan may be easily encoded in a IMS-LD, packaged in a IMS-CP that contains all the involved learning objects and delivered for execution in most LMS. The procedure described along this paper has been implemented in Python and fully integrated in the ILIAS LMS [9], which embeds a SOAP (Simple Object Access Protocol) server, so that several Python scripts implement the extraction procedures described so far, just by using the available SOAP functions, and obtain the domain and problem files. The SIADEX HTN planner [3] is then executed and a plan is obtained. ILIAS does not support IMS-LD specification yet, so in order to make the plan available to student, we have translated the plan into a follow up guideline that appears over the student' ILIAS desktop.

## 5 Related work

There are several approaches in the literature that shows the relative success of AIP&S technologies for the assisted design of learning paths, either for HTN planners [2] or non HTN planners [8]. In all these approaches, the planning domain cannot be encoded by the instructor, but by a person with deep knowledge on AIP&S and PDDL. This means that any change in the learning objects repository has to be recoded again, making the instructor to depend on third persons. Our approach clearly grants the independence of instructors and reduces the cost of using AIP&S to zero.

## 6 Concluding remarks

This paper has presented an integrated approach able to extract a planning domain from a well structured learning objects repository just by exploring the standard metadata labeling present in the objects. This is just a first step towards the automatic use of an HTN planner able to obtain customized learning paths adapted to every student' needs and features. The use of an artificial intelligence HTN planner allows for a fast and robust generation of adapted learning designs with respect to typical learning designs that had to be manually encoded in a long and boring process. The main obstacle for the practical use of AI HTN planners, i.e. the design of the planning domain, has also been overcome without any intervention of instructors.

However there is still an issue that needs further study: the adaptation of the learning path to run-time information like the result of intermediate evaluations during the development of a course. We are pursuing a continual planning approach [7] in which the planning of a whole course with intermediate evaluations is neglected in favour of a sequence of planning episodes, each planning episode happening between every two consecutive evaluations. It is clear that if a course has no intermediate evaluations, then the continual planning approach is not necessary and a unique plan is enough to cover the whole development of a course.

## References

1. ANSI/IEEE. IEEE Standard for Learning Object Metadata. `http://ltsc.ieee.org/wg12/`.
2. Ullrich C. Course Generation Based on HTN Planning. In *Proceedings of 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, pages 74–79, 2005.
3. L. Castillo, J. Fdez-Olivares, O. García-Pérez, and F. Palao. Efficiently handling temporal knowledge in an HTN planner. In *Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*, 2006.
4. M. Ghallab, D. Nau, and P. Traverso. *Automated planning: theory and practice*. Morgan Kaufmann, 2004.
5. IMS-GLC. IMS Global Learning Consortium. `http://www.imsglobal.org/`.
6. D. Long and M. Fox. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.
7. K. L. Myers. CPEF: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, 1999.
8. M.D. R-Moreno and David Camacho. AI techniques for Automatic Learning Design. In *International electronic Conference on Computer Science (IeCCS-2006)*, 2006.
9. ILIAS Learning Management System. ILIAS website. `http://www.ilias.de/ios/index-e.html`.
10. D. E. Wilkins and M. desJardins. A call for knowledge-based planning. *AI Magazine*, 22(1):99–115, 2001.