# Using AI Planning to Enhance E-Learning Processes

**Antonio Garrido**
Universitat Politecnica de Valencia
Camino de Vera s/n
46022, Valencia (Spain)

**Lluvia Morales**
Universidad Tecnologica de la Mixteca
Carretera a Acatlima Km. 2.5
69000, Huajuapan de Leon, Oax. (Mexico)

**Ivan Serina**
Free University of Bozen-Bolzano
viale Ratisbona 16
39042 Brixen-Bressanone (Italy)

**Abstract**

This work describes an approach that automatically extracts standard metadata information from e-learning contents, combines it with the student preferences/goals and creates PDDL planning domains+problems. These PDDL problems can be solved by current planners, although we motivate the use and benefits of case-based planning techniques, to obtain fully tailored learning routes that significantly enhance the learning process. During the execution of a given route, a monitoring phase is used to detect discrepancies, i.e. flaws that prevent the student from continuing with the original plan. In such a situation, an adaptation mechanism becomes necessary to fix the flaws, while also trying to minimise the differences between the original and the new route. We have integrated this approach on top of *Moodle* and experimented with 100 benchmark problems to evaluate the quality, scalability and viability of the system.

Most everyday, real-world activities imply some planning to determine a series of tasks to reach certain goals under definite constraints. A good proof of this is reflected in the last ICAPS International Planning Competitions, which put special emphasis on domains that are (desirably) related to real applications. The last objective pursued in planning technology is not *simply* to solve problems efficiently, but also to bridge the gap between its techniques/algorithms and the AI-illiterate final users, who do not usually possess great skills in PDDL planning and computer knowledge.

This work describes an approach, named *myPTutor*, which takes as an input an e-learning model described in a standard e-learning language and produces a solver-ready PDDL model as an output. In particular, it applies standard AI planning and Case-Based Planning (CBP) techniques to the generation of fully tailored e-learning routes. This approach contributes with a full vision, which includes: i) the extraction of metadata information from Learning Objects (LOs) encoded in e-learning standards; ii) the automated compilation of standard PDDL domain+problem files, which represents a knowledge engineering stage; iii) the generation of a learning route, i.e. plan, by a case-based planner or other planner; iv) the execution and monitoring of such a route within *Moodle*, a well-known Learning Management System (LMS); v) the adaptation of the learning route where a discrepancy between the real and the expected state appears, i.e. a repair method to fit the new scenario; and vi) all of this in a transparent-to-the-user way, who does not need to worry about the technical e-learning or planning insights.



## Problem Description

The Web is full of interoperable digital resources, known as LOs, implemented in XML standards such as SCORM, IMS or LOM (LOM 2002; SCORM 2004). But LOs in themselves are insufficient to accommodate the different studying styles and preferences of the students in large courses. The right selection and combination of LOs within these courses is the basis for e-learning, a multi-disciplinary field to facilitate and enhance learning. For instance, according to pedagogical issues, a lecture is very recommendable for verbal students but not for visual ones, and just the opposite holds for a diagram. Consequently, e-learning must provide a student-centered solution by offering a learning process where courses are tailored to the specific needs, learning styles, background and, in general, profile of each student.

After generating a tailored learning route, it needs to be executed in a real LMS. This means to assist students in navigating the route, monitoring it, checking its current progress and acting when discrepancies, that is differences w.r.t. the expected state, appear. If this happens, a flexible process to make the remaining learning route executable (by adding or removing LOs) becomes essential. And it is expected that this process will not ignore the original students' interests, but try to reuse the original route as much as possible.

## Related Work. Motivation for Using Planning

Many techniques have been traditionally applied to generate individualised courses in e-learning, such as adjacency matrices, integer and constraint programming models, neural networks and soft computing methods (Brusilovsky and Vassileva 2003; Garrido, Onaindia, and Sapena 2008; Idris, Yusof, and Saad 2009; Martinez et al. 2004). They simulate human decision-making, which has a disadvantage because the flow of LOs is somewhat predefined and, consequently, too teacher-oriented. AI planning related works have been used to provide students with learning routes based on their

preferences (Castillo et al. 2010; Kontopoulos et al. 2008; Limongelli et al. 2009; Ullrich and Melis 2009), but their main limitations are: i) they do not fully use e-learning standards, ii) they are not displayed and integrated in widely used LMSs, and iii) they are usually limited to a specifically designed ontology and/or planning paradigm. On the contrary, our approach lets any type of planner to find the best flow of LOs according to pedagogical theories, with the idea of bringing the right content to the right student, and also deals with standard e-learning metadata, which is automatically extracted and compiled as a PDDL model. Additionally, it allows teachers to define compulsory or optional goals, an extra feature in many LMSs.

Metaphorically speaking, generating a learning route resembles planning closely. In e-learning the main elements are: i) the students' background/preferences, ii) the learning goals to be attained, iii) the profile-adapted LOs with their prerequisites and learning outcomes, iv) the ordering relations, and v) the tailored learning route. We map these elements into planning terms, respectively, as: i) initial state, ii) top level goals, iii) actions with preconditions and effects, iv) causal link relations, and v) the solution plan. Planning also deals with multi-criteria optimisation, very appealing for e-learning: students (and teachers) prefer a good learning route, in terms of time, competence, resources or cost, and not simply yet another route.

Focusing on planning algorithms, CBP takes advantage of former problem-solving experiences by storing in a plan library previously generated solutions that can be reused to solve similar problems in the future (Serina 2010). This is essential in e-learning. First, teachers want to include didactic issues that cannot be represented in terms of preconditions, effects and causal links. Although PDDL3 plan trajectory constraints and preferences may be used here, in most cases teachers cannot formally express why a route or a LO is better than other. And when they can, they are reluctant to use complex and tedious constructs that are unfamiliar to them. Second, if the execution of a learning route fails it needs to be fixed. Although a new route starting from scratch can be replanned, it does not seem to be sensible; a route adaptation that minimises the differences between the original and the recalculated route seems more reasonable. Informally, students and teachers prefer a kind of *inertia* in the learning routes, which enhances the continuity in the learning process. CBP techniques are very valuable in this scenario for the definition, memorization (of alternatives which are motivated by reasons beyond causal links and orderings), retrieval and adaptation of learning routes, i.e. plans.

## Our approach: *myPTutor*

### Architecture and Workflow

*myPTutor* has a mixed-initiative architecture for both teachers and students, as depicted in Figure 1. Initially, teachers (in the role of course designers) define the course by creating the LOs or reusing them from available repositories. After modelling the student's profile (e-portfolio with the background, learning styles and interests), an automatic translator compiles all this information as a PDDL domain+planning problem. When the planner generates a plan (i.e. learning route), it can be validated by a teacher and stored in the CBP library. This tailored route is uploaded to the LMS as a plan manifest that allows a student to navigate through it. The LMS also monitors the execution of each route and if any discrepancy is found between the real and expected state, a new planning iteration is launched to adapt the learning route. We now detail this architecture.

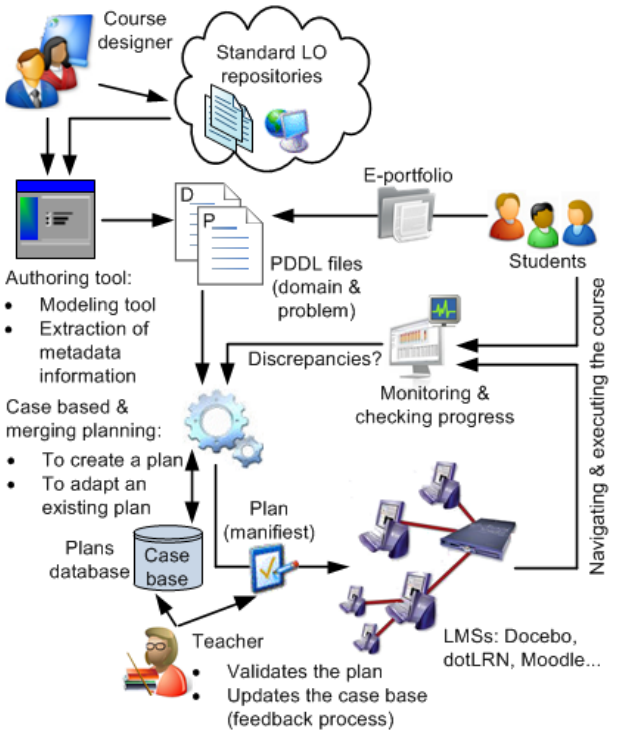



Figure 1: Architecture of *myPTutor* for using planning in an e-learning setting. More info in `http://servergrps.dsic.upv.es/myptutor`.

### Definition of the Course

According to (Polsani 2003), *"as individual words cannot independently produce meaning, the LOs in themselves are insufficient to generate significant instruction [. . .] How many LOs, how they are related, and for what purposes will be determined by the instructor's objectives, pedagogical methodology and instructional design theories."* This summarises the necessity to define a course by relating LOs, which can be done by using existing editors that provide textual templates to fill in the contents, such as RELOAD (`http://www.reload.ac.uk`) or eXeLearning (`http://exelearning.org`). In addition to these, we have implemented a graphical tool that simplifies the course definition; it allows drag&drop of visual components (LOs, their relationships and profile adaptation) in an intuitive way, analogously to a concept map, as shown in Figure 2. This tool is also planning oriented, representing the preconditions/effects (circled elements) for each action (squared elements), which facilitates the definition of

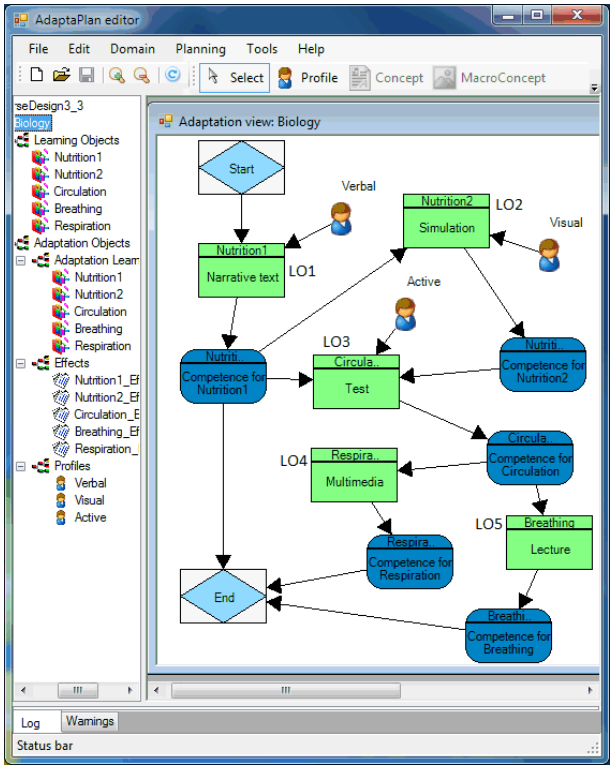



Figure 2: Graphical definition of a short course on Biology that includes: LO1 (Nutrition1), LO2 (Nutrition2), LO3 (Circulation), LO4 (Respiration) and LO5 (Breathing). Three profiles (verbal, visual and active) are also modelled.

the course and, indirectly, the planning domain that is subsequently generated. The objective is therefore twofold. For average users, to offer a very simple way to reuse (or define) classical LOs within a course. For advanced users, to provide richer possibilities to extend these LOs with more complex capabilities, such as conjunctive/disjunctive/recommended requirements, minimal/maximal duration allowed per LO, number of students needed for collaborative work, resources to be used and their cost, metric to be optimised, etc.

## Compilation of the PDDL Model

We use a knowledge engineering method, based on (Garrido et al. 2009; 2012), to extract the metadata information and compile the set of LOs as the PDDL domain. The LOM specification for metadata has many pedagogical entries, but just three of them are essential to support planning:

- Relations and their kind, which represent the content dependencies among LOs; e.g. *Requires*, *IsBasedOn* and *References* constructs, which mean conjunctive, disjunctive and soft/recommended relations, respectively.
- Educational information and LO's type, e.g. a lecture, narrative text, diagram, etc., which gives us an idea of the difficulty and duration of the LO. According to education experts, the type of the LO has a positive or negative interaction with the student's profile.
- Technical requirements, which represent the particular resources needed by the LO; e.g. a multimedia device or specific operating system. This entry tends to be rather vague and it does not always represent a resource as classically defined in planning terminology. In such a case, which is up to the LO designer, this entry is just ignored.

The compilation of the PDDL domain is an automated polynomial process that iterates all over the LOs and generates one planning operator per LO. This generation relies on both a sound definition of the metadata and a closed world assumption. First, if the LO designer fails in the metadata labelling (e.g. *"LO1 Requires LO2"* and *"LO2 Requires LO1"*), which surprisingly is not uncommon, the domain will be incorrect, though this can be easily detected by domain analysis tools. Second, if the LO changes, its corresponding operator must be recompiled (but others operators do not need to change). In short, each operator consists of:

- A unique name taken from the LO name/identifier.
- One parameter, the student, to support the personalisation.
- The duration, as the LO learning time. Note that this item can be ignored if we do not use temporal domains.
- The preconditions, to filter the students' profile and to support the dependency relations. Other elements, such as the students' role, technical or educational requirements can be also modelled as preconditions.
- The effects, to represent the LO outcome. Other elements, such as optional expressions on rewards w.r.t. the student's profile, resource costs or additional metric values can be easily included.

According to this description, the PDDL durative action for LO3 of Figure 2 is:

```
(:durative-action LO3 ;Circulation (Test)
 :parameters (?s - student)
 :duration (= ?duration 2) ;LO typicalLearningTime
 :condition (and
  (at start (not (LO3 ?s done))) ;avoids repetitions
  (over all (profile ?s active)) ;profile dependency
  (at start (LO1 ?s done))  ;Requires LO1, Nutrition1
  (at start (LO2 ?s done))) ;Requires LO2, Nutrition2
 :effect (and
  (at end (LO3 ?s done)) ;competence achieved
  (at end (increase (reward ?s (profile-active))))))
```

The compilation of the PDDL problem models the initial state, the goals and, when necessary, the metric to optimise, all directly extracted from the students' e-portfolio. The initial state represents the students' profile and background (e.g. learning style, initial background on nutrition, etc.), the preferred language of the course, and some other information (e.g. special equipment or resource availability). The goal is to pass the entire course or a part of it, usually in terms of some LOs (e.g. `(LO5 student1 done)`).

## Solving the Planning Problem via CBP

Similarly to other Case-Based Reasoning (CBR) systems, CBP is based on two assumptions on the nature of the world (Leake 1996). First, the world is regular: similar problems

*Algorithm* UPDATELIBRARY
*Input*: A solution plan $\pi$ for $\Pi = (I, G)$ and a set of facts $F$;
*Output*: Update the plan library inserting new elements obtained considering subplans of $\pi$;
1. compute the set of causal links $\mathcal{C}_\pi$ in $\pi$;
2. $S = G \cup F \cup \{G_j \subseteq G \mid \bigcap_{g_j \in G_j} \pi_{g_j} \neq \emptyset\}$;
3. `forall` $G_i \in S$ `do`
3.1 CHECK&INSERT$(\Pi_{G_i}, \pi_{G_i})$;

Figure 3: Algorithm for updating the plan library inserting subplans of a given input plan $\pi$.

have similar solutions. Second, the types of problems an agent encounters tend to recur; hence future problems are likely to be similar to current problems.

To the end of applying the reuse technique, it is necessary to provide a plan library from which "sufficiently similar" reuse candidates can be chosen. In this case, "sufficiently similar" means that reuse candidates have a large number of initial and goal facts in common with the new instance. However, one may also want to consider the reuse candidates that are similar to the new instance after their objects have been systematically renamed. This corresponds to identifying a mapping between the objects of the reuse candidate and the objects of the new instance, such that the number of common goal facts is maximised and the additional planning effort to achieve the initial state of the plan library is minimised. This is extremely important in e-learning, where teachers could decide to reuse a course, or a part of it, that has been previously adopted by their colleagues or by themselves. Obviously, the students will not be the same, but if their profile, their goals and the resources available are similar to the corresponding ones in the case base, our system will be able to propose a new high quality plan with a limited number of changes w.r.t. the one stored in the library.

Our CBP approach is built on top of the OAKPLAN system (Serina 2010), which uses an approximate evaluation based on kernel functions to compute an appropriate mapping between the students and the objects of the reuse candidate and the corresponding ones of the current instance, which can be computed in polynomial time. In order to improve the efficiency of the system and reuse as many possible parts of previously executed plans we have adopted *plan merging techniques* (Yang, Nau, and Hendler 1992). These plans can be generated by teachers, or by a planner (and then validated by teachers). In particular, we decompose the solution plans into subparts that allow us to satisfy every single goal, or a set of interrelated goals, and then we store these subparts in the case base, if they are not already present.[1]

Figure 3 describes the algorithm for updating the plan library with parts of an input plan $\pi$. In short, UPDATELIBRARY identifies the subplans of $\pi$ that can be inserted in the plan library to increase the *competence* of the library in itself (Smyth and McKenna 2001; Tonidandel and Rillo 2002). Here $\pi_g$ represents the subplan of $\pi$ that satisfies $g$ starting from $I$. Note that it can be easily identified considering the set of causal links $\mathcal{C}_\pi$ of $\pi$ computed at step 1. In a similar way, it is possible to compute the set of facts $I_g$ that are necessary to apply the actions of $\pi_g$.

At step 2 we identify the set of facts that will be examined for the insertion in the library. In particular we consider all the goals $G$, the elements of $F$ and the subsets of *interacting* goals $G_i$. The $F$ set represents a set of facts, different by the input goals, that could be useful for the following merging phase such as unsupported facts of a previous adaptation phase. Moreover, the sets of interacting goals $G_i$ can be easily computed considering the actions in the subplans $\pi_{g_j}$ that are in common to the different goals.

The CHECK&INSERT$((I_{G_i}, G_i), \pi)$ function (step 3.1) searches if there not exists a case-base element $(\Pi_j, \pi_j)$ whose goals and initial state perfectly match with the current goals and initial state, respectively. In this case, we insert the current planning problem $\Pi_{G_i} = (I_{G_i}, G_i)$ and its solution plan $\pi_{G_i}$ in the library. Otherwise, we have to decide whether to insert $(\Pi_{G_i}, \pi_{G_i})$ and remove $(\Pi_j, \pi_j)$, or simply skip the insertion of $(\Pi_{G_i}, \pi_{G_i})$. In our tests we have used an update policy that maintains the plan with the lowest number of actions, but other policies could be used as well considering, for example, the plan qualities, their makespan, or the robustness to exogenous events. Moreover, CHECK&INSERT ignores too small and too big plans $\pi$; in fact, a small plan $\pi$ could determine the inclusion in the library of a high number of very small plan fragments that have to be considered in the merging phase, while a big plan $\pi$ could determine the insertion in the library of very big subplans that are difficult to merge.[2]

When a new e-learning planning problem must be solved, we search in the case base if a plan that already solves all goals exists. If such a plan does not exist we apply plan merging techniques that progressively identify (sub)plans in the case base that can satisfy the goals. This phase consists in reusing parts of the retrieved plans to complete a new one. Figure 4 describes the process for merging plans of the library in order to find a plan $\pi$ that solves the current planning problem $\Pi$ or that represents a quasi-solution (Gerevini and Serina 1999) for it. At step 1 we search in the library the plan that satisfies all the goals with the lowest heuristic adaptation cost, where the function $EvPlan(I, \pi, G)$ determines the adaptation effort by estimating the number of actions that are necessary to transform $\pi$ into a solution of the problem $\Pi(I, G)$.[3] This step corresponds to the extraction of the best plan of the library (if it exists) as proposed by the standard OAKPLAN system (Serina 2010). At steps 3.x, we progressively analyse the unsatisfied goals and the unsatisfied preconditions of the current plan $\pi$, trying to identify in the library a subplan $\pi_f$ that can be merged with $\pi$ in order to satisfy $f$ (and other unsatisfied facts if possible) and reduce, at the same time, the global heuristic adaptation cost, where $merge$ identifies the best part of $\pi$ where the actions of $\pi_f$ can be inserted in producing a new global plan.[4] If such a

[1] See (Roubickova and Serina 2012) for a detailed description of this approach.

[2] In our tests we have used $5 \leq |\pi| \leq 200$.

[3] See EVALUATEPLAN in (Serina 2010) for a more detailed description.

[4] In our tests we have considered the earliest and the latest part of $\pi$ where $f$ can be satisfied.

*Algorithm* MERGESUBPLANS
*Input*: A planning problem $\Pi(I, G)$, a plan library $\mathcal{L} = (\Pi_i, \pi)$;
*Output*: A (quasi) solution plan $\pi$ for $\Pi$;
1. $\pi = argmin_{\pi_i \in \mathcal{L} \cup \emptyset} EvPlan(I, \pi_i, G)$;
2. `repeat`
3. `forall` unsatisfied facts $f \in \{G \cup prec(\pi)\}$ `do`
3.1 Let $\pi_f \in \mathcal{L}$ be the best plan that satisfies $f$ s.t. $EvPlan(I, merge(\pi, \pi_f), G) < EvPlan(I, \pi, G)$;
3.2 `if` $\pi_f \neq \emptyset$ `then`
3.3 $\pi = merge(\pi, \pi_f)$; break;
4. `until` $\pi_f \neq \emptyset$;
5. `return` $\pi$;

Figure 4: Algorithm for merging the elements in the library in order to solve a planning problem $\Pi$.

plan exists, we merge it with $\pi$ at step 3.3 and we restart from step 3 reconsidering all the unsatisfied facts. The repeat loop halts when all the goals and preconditions are satisfied, i.e. when we have found a solution plan, or when there is not a suitable plan that can be extracted from the library that satisfies the remaining unsupported facts. In this case, the plan $\pi$ does not represent a solution plan. However, it can be used as a starting point for a local search process to find a solution plan for the current planning problem.

This way of proceeding allows teachers to easily validate the proposed learning route. They can simply check the parts of the route that differ from the elements stored in the case base and that have been introduced to satisfy, for example, new students' goals or prerequisites, instead of reconsidering the whole learning route. Note that different criteria can guide the definition of a learning route. In our approach we do not only try to find good quality plans that best fit the students' requirements, but also to minimise the number of LOs that have been introduced or removed w.r.t. the case base elements. The relative importance of plan quality w.r.t. plan stability (Fox et al. 2006) can be chosen by the teacher when the case-based planner is executed. In our context, the *learning route plan stability* is measured considering the *distance*, expressed in terms of number of different LOs, between the source learning route and the target learning route.

## Execution and Monitoring

Once the learning route has been generated, we embed its sequence of LOs in a learning manifest, which can be executed in any LMS that integrates a SCORM player. During this execution many situations can happen. First, discrepancies between the expected state (obtained by a flawless execution of the proposed plan) and the real state (resulting from the current execution of the plan) may arise, such as when the student does not pass a critical evaluation task, or when a LO with dependencies takes more time than initially scheduled. Second, the students' profile can change: increment/decrement in their performance levels according to previous evaluation tasks, improvement in their foreign language levels, acquisition of new resources (e.g. multimedia equipment or software), etc. These situations, and others that can be defined in the future, are automatically recovered from the LMS database —which includes the student's e-portfolio—, as it is immediately updated after an evaluation activity is finished and/or graded. Note that we do not perform a continuous monitoring since it may become inviable in an e-learning scenario where each student logs in, and off, frequently and works at his/her own pace, but only after evaluation tasks, which are usually graded by a teacher.

Technically, in our implementation we create a new planning problem where the students' initial state is the current state and the learning goals remain the same (although they can be changed if desired). The original domain, the new planning problem, and the remaining part of the plan that is yet to be executed are provided to the planning server, which uses a plan validator (*VAL*idation tool, `http://planning.cis.strath.ac.uk/VAL`) to check whether that plan is still executable. If it contains flaws, the CBP is invoked in order to find, or adapt, a new plan for the given student under the current scenario. The entire part of the plan is provided to the CBP, which increases the possibilities to fix the flaws with minimal modifications.

## Adapting the New Plan

The plan adaptation system consists in reusing and modifying previously generated plans to solve a new problem and overcome the limitations of planning from scratch. Similarly to OAKPLAN, our work uses the LPG-ADAPT system (Fox et al. 2006) given its good performance in many planning domains, but other plan adaptation systems can be used as well. LPG-ADAPT is a local-search-based planner that modifies plan candidates incrementally in a search for a flawless candidate.

Any kind of planning system that works in dynamic environments has to take into account failures that may arise during plan generation and execution. In this respect, CBP is not an exception. This capability is called plan revision and it is divided into two subtasks: evaluation and repair. The evaluation step verifies the presence of failures that may occur during plan execution when the plan does not produce the expected result. When a failure is discovered, the system reacts by looking for a repair or aborting the plan. After finding the plan in the library and repairing it with the plan adaptation techniques, the solution plan can be inserted into the library or discarded.

## Putting all Together. Integration with *Moodle*

Our architecture has been designed to be flexible, thus being compatible with any LMS, and to support any PDDL planner. In order to validate our approach, we have implemented it on top of *Moodle* (`http://moodle.org`), maintaining its original core code. Although the technical details are beyond the scope of this paper, we have implemented several extensions to allow a mixed-initiative mechanism between users (students and teachers) and the planning services.

First, we have defined new tables in the relational database to support planning preconditions, the students' background (i.e. the initial state), the goals that are compulsory for each course and those that can be optionally chosen by each student. Second, new forms have been added to the teacher's GUI to define the course goals (compulsory and optional) and the initial profile required for the students (see Figure 5-1). Similar forms have been designed for students to input their profile information and desired optional goals.

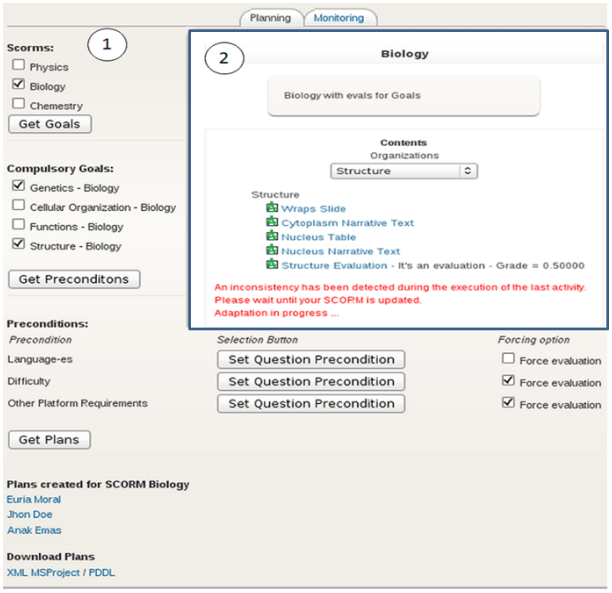



Figure 5: 1): module that allows teachers to select the course, its goals, background preconditions and adapt SCORMs to different students' profiles through planning services. 2): monitoring during the student SCORM execution.

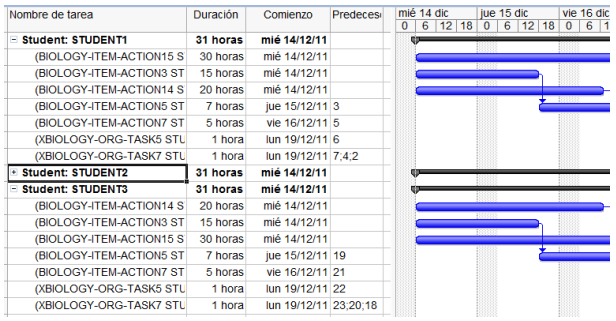
Once this information has been input into the system, it automatically invokes the planning Web service to calculate a personalised learning route per student (Figure 5-2).

Finally, other forms are designed to execute and monitor the state of the learning route. *myPTutor* also outputs the learning route as a Gantt chart in Microsoft Project format (see Figure 6). From the teacher's point of view, (s)he can observe this chart of expected LOs and manually compare it with the "already done" sequence of LOs. From the student's point of view, this chart shows the schedule of the next LOs to be executed. Again, if during the student execution of the learning route the system detects discrepancies between the expected and real state, a message is displayed to the student, as depicted in Figure 5-2. This message indicates the remaining LOs that cannot be executed from the current state, and the process stops until the system performs an adaptation of the route. This process continues until the learning route is completed and all goals are achieved.

## Evaluation and Experimental Results

A thorough evaluation of this approach is not easy, as it involves the strong collaboration of high numbers of educational researchers for a correct definition of the courses and for a comprehensive testing with real students. Just to shed some light on the validity of the approach, we present both a qualitative and quantitative evaluation.

We address the qualitative evaluation by means of opinion questionnaires answered by a group of 10 teachers and 10 students to assess the learning routes, their size/duration and their adequacy, in terms of their LOs, to the particular profiles. This is still ongoing work because not many teachers are willing to participate in this type of personalised approach —they prefer continuing with their classical way of teaching. Preliminary results show that the teachers agree with the routes in terms of their form, size and adaptation to the students. But teachers also recognise that it is hard to evaluate a learning route to each profile (Garrido et al. 2012). As for students, the experience was highly positive. They found the provided LOs a very helpful way to catch up with the background required for the course, and an ideal mechanism for self-assessment. Also they found the course was specifically designed for them, and not "the same course for everybody".



| Nombre de tarea | Duración | Comienzo | Predeces |
|---|---|---|---|
| Student: STUDENT1 | 31 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION15 S | 30 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION3 ST | 15 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION14 S | 20 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION5 ST | 7 horas | jue 15/12/11 | 3 |
| (BIOLOGY-ITEM-ACTION7 ST | 5 horas | vie 16/12/11 | 5 |
| (XBIOLOGY-ORG-TASK5 STU | 1 hora | lun 19/12/11 | 6 |
| (XBIOLOGY-ORG-TASK7 STU | 1 hora | lun 19/12/11 | 7;4;2 |
| Student: STUDENT2 | 31 horas | mié 14/12/11 | |
| Student: STUDENT3 | 31 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION14 S | 20 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION3 ST | 15 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION15 S | 30 horas | mié 14/12/11 | |
| (BIOLOGY-ITEM-ACTION5 ST | 7 horas | jue 15/12/11 | 19 |
| (BIOLOGY-ITEM-ACTION7 ST | 5 horas | vie 16/12/11 | 21 |
| (XBIOLOGY-ORG-TASK5 STU | 1 hora | lun 19/12/11 | 22 |
| (XBIOLOGY-ORG-TASK7 STU | 1 hora | lun 19/12/11 | 23;20;18 |



Figure 6: Gantt chart that shows the learning routes of three students.

The quantitative evaluation is slightly easier. The compilation of a standard PDDL model facilitates the use of any modern planner and, therefore, we can run many computational experiments. In particular, we test the effectiveness of our case-based approach with merging techniques, implemented on top of OAKPLAN (Serina 2010), which we have called OAKPLAN-merge, *vs.* plan generation techniques when discrepancies appear while executing the learning routes. Particularly, we focus on: i) the scalability by means of the CPU time required to obtain (retrieve&adapt or replan) the routes, ii) the best quality solutions in terms of higher reward plans, and iii) the best stability that can be obtained in a given deadline.

We have experimented with several courses, but here focus on a real, large-size *Moodle* course of around 90 LOs on Natural Sciences inspired on `http://www.profesorenlinea.cl`. We have created nine initial configurations (with 10, 20...90 fictitious students, respectively), and defined 10 variants per configuration (plus one additional variant for the 90-th problem), thus considering 100 planning problems in total (the 91 variants plus the 9 initial configurations). Each variant artificially simulates the changes that may occur during the route execution in an incremental way. That is, in the first variant some equipment is no longer available. The second variant maintains these changes and includes restrictions on the students' availability; and so on for the other variants.

In addition to OAKPLAN and our case base planner OAKPLAN-merge, we have used two state-of-the-art planners, SGPLAN6 and LPG.[5] All tests were performed on an

[5]For a further description of these planners see

Intel(R) Xeon(TM) CPU 2.40GHz with 2GB of RAM, and censored after 10 minutes. In our tests, the solution plans (i.e. the learning routes) inserted in the case base were obtained by using the best quality plans generated by LPG and SGPLAN6 on the initial-configuration planning problems used to create the corresponding variants.

Figure 7 depicts the results required by the different planners. On the left we compare OAKPLAN (with and without merging techniques) *vs.* LPG and SGPLAN6 using a "complete" case-base that contains all the base problems and the corresponding solutions (the case-base for the merging variants contains also the selected subplans of the base problems). Here we can observe the general good behaviour of the case-based techniques, which are comparable in terms of CPU-time to SGPLAN6. The results show that the case-based approach is at least as fast as replanning, and sometimes faster. Obviously, plan retrieval techniques show less useful when the changes are significant and fixing the route requires more effort than simply discarding it and rebuilding a new one from scratch. But, as Table 1 shows, the benefits for investing this effort pay off in terms of stability.

In Figure 7-right, we analyse the behaviour of OAKPLAN and OAKPLAN-merge to study the impact of using a case base considering: i) a case base created using only the smallest base problem (with 10 students), ii) a case base where the base problems are progressively inserted *after* the corresponding variants have been evaluated (it initially contains only the smallest base problem). In the first case, we primarily want to evaluate the ability of the merging techniques to reuse the solutions available in the case base at the increase of the "differences" (in terms of number of students) among the current situation and the elements stored in the case base. In particular, we want to examine the scalability in terms of number of students, which is extremely important in our context since a teacher could decide to evaluate the effectiveness of an e-learning course considering a limited number of students before using it for the whole class.

Considering the tests with the *small* case base, we can observe the general good behaviour of OAKPLAN-merge, while OAKPLAN *without* merging techniques is able to solve only 50 variants. Regarding the tests with the *incremental* case base, we want to analyse the behaviour of OAKPLAN considering a case base that contains elements which are structurally not too much different w.r.t. the current situation. For example, considering the solution of the variants with 50 students, the case base does not contain the base problem with 50 students but it contains the base problems with 10, 20, 30 and 40 students. Here we want to examine the situation where a teacher has already used a course in different classes and wants to reuse the stored experiences in a new (slightly bigger) class. As expected, the behaviour of OAKPLAN-merge does not change significantly. On the contrary, the CPU-time of OAKPLAN *without* merging techniques decreases significantly since it can replan starting from a case base element with a slightly lower number of students w.r.t. the current situation. Moreover, it is now able to solve all the variants considered.

`http://ipc.icaps-conference.org`.

| Planner | Num | Speed | Quality | Distance |
|---|---|---|---|---|
| OAKPLAN | 100 | 16.31 (66.8) | 72098 (**99**) | 18.76 (12) |
| small CB | 50 | 13.79 (30.6) | 38973 (47.4) | 139 (0.63) |
| incr. small CB | 100 | 13.5 (**94**) | 71048 (97.3) | 233 (0.91) |
| OAKPLAN-merge | 100 | 30.66 (41.7) | 70444 (95.9) | 2.06 (80.8) |
| small CB | 100 | 40.3 (32) | 68709 (93.5) | 2.18 (**86.4**) |
| incr. small CB | 100 | 35.8 (37.6) | 69607 (94.5) | 2.21 (85.7) |
| SGPLAN6 | 100 | 15.73 (84.8) | 69883 (95.4) | 231.5 (0.76) |
| LPG | 66 | 67.46 (9.3) | 48315 (63.2) | 163 (0.87) |

Table 1: Number of problems solved, average CPU-time (seconds), average Quality and average Distance of OAKPLAN, OAKPLAN-merge, SGPLAN6 and LPG. The corresponding IPC scores are reported in the round brackets.

Table 1 summarises the experimental results for the different planners considered in terms of number of solved problems, average total CPU-time in seconds, plan quality and plan distance. Note that we want to maximise the total reward of the students and so the higher the quality, the better the plan is. Here we can observe the general good behaviour of the case-based techniques, which is particularly evident considering the IPC scores reported in brackets. Briefly, each planner receives a score between 0 and 1 for each problem solved. The score is the ratio between the time/quality/distance value of the solution found and the time/quality/distance value of the best solution found by any other planner. The score is summed across all problems for a given planner; the higher the score, the better the planner is. In particular, OAKPLAN-small with incremental case-base is the fastest, followed by SGPLAN6. Standard OAKPLAN produces the best quality plans, and OAKPLAN-merge and SGPLAN6 perform extremely well. Regarding the plan distance, we can observe the good behaviour of OAKPLAN-merge with values extremely different w.r.t. the other planners. Note that the retrieval and adaptation process sometimes comes at a price in terms of quality, as the route is adapted to fit a new configuration rather than constructed expressly for it. But our experiments show that the quality for the case-based approach can be better than for replanning, particularly in the most complex problems.

The best values for plan distance are achieved in OAKPLAN-merge. While replanning generates routes that are consistently very different from the original ones, the differences between the retrieved plan and the solution plans are very small. This is not particularly surprising since LPG and SGPLAN6 do not know the target plans used for this comparison. These distance values are interesting since they are a clear indicator of the good behaviour of case-based techniques and show that the generative approach is not feasible when we want to preserve the stability of the plans produced. We can also observe the extremely good behaviour of OAKPLAN-merge. Its distance values are obtained considering the number of different actions w.r.t. the matching plan provided by the retrieval phase, which is not necessarily obtained directly by a single solution plan stored in the case base (as in OAKPLAN), but also using the different subplans (highlighted to the teachers) obtained by the analysis of the case-based elements that best fit the current goals and initial

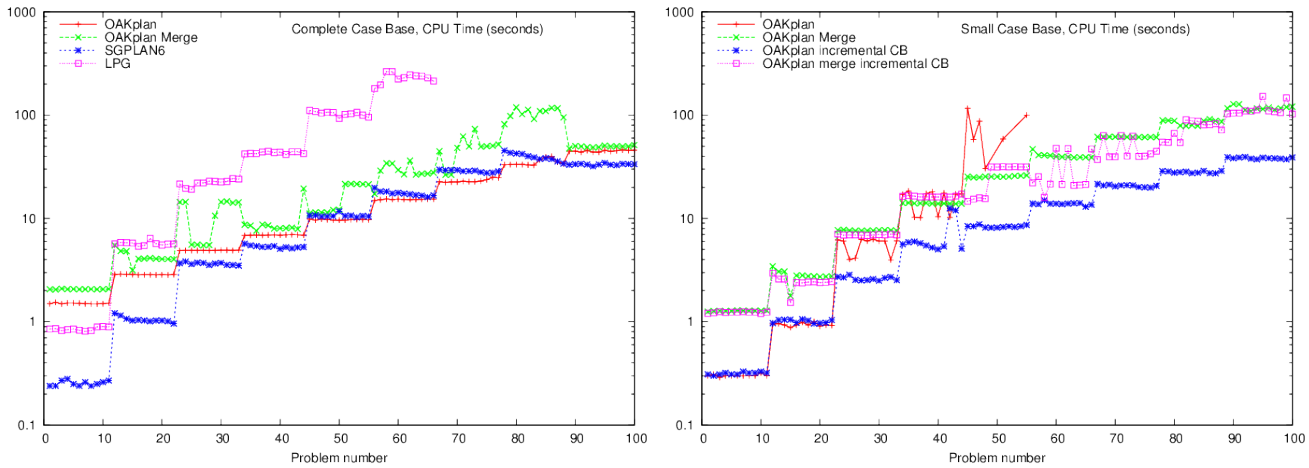



Figure 7: CPU time in seconds (on a logarithmic scale) required by the different planners in order to solve our benchmark domains. In the plot on the left we compare the case-based approaches (OAKPLAN and OAKPLAN-merge) *vs.* replanning (LPG and SGPLAN6). On the right we compare the case-based approaches considering different input case bases.

state. This indicator is very appealing in an e-learning setting as the students/teachers do not want to deal with an entirely new learning route after a little change happens during execution. Quite the contrary, students and teachers prefer a kind of *inertia* in the (already known) learning routes.

Regarding the plan differences produced by OAKPLAN *without* merging techniques w.r.t. the solution of the base problems we can observe values which are similar to the ones obtained for replanning. This is not surprising since the elements stored in the small and in the incremental case bases are very different w.r.t. the new planning problems. On the contrary, the performance of OAKPLAN-merge is extremely good, both in the small and incremental case bases. However, it is important to point out that in this case the comparison in terms of plan distances are performed considering directly the plan provided by the retrieval phase. It is up to the teacher to decide if (s)he wants to validate elements that only deal with previously executed courses or also subparts of them.

Globally, we can observe that the use of plan merging techniques is potentially very effective in our context, allowing us to obtain efficiently new e-learning routes which are very similar to the combination of previously executed ones (or subparts of them). This is extremely important since it allows to highlight the teachers the changes w.r.t. already executed learning routes, facilitating the validation process. Furthermore, the stored plans can also contain some notes, regarding for example the pedagogical motivations associated to the selection or combination of specific LOs, annotated by the teacher during the creation of the original learning route, or during previous executions of the learning route. In this case, these notes can be easily reexamined by the teachers facilitating the learning route validation process.

## Conclusions and Lessons Learnt

We have described a flexible approach for personalisation of e-learning routes that integrates techniques from different fields, such as educational theories, profile identification and modelling, knowledge representation, planning algorithms and optimisation procedures.

The key lesson learnt is that planning technology must be introduced transparently to the user. Final users, in our case teachers and students, do not want to deal with PDDL constructs. On the contrary, they want to continue using their courses, LOs and LMSs with a minimal effort. And here an automated knowledge engineering compilation based on LOs' metadata demonstrates very useful. But this metadata definition is sometimes tricky, as it is not always fully specified. Consequently, this is still a challenging issue: to find LO repositories designed to be interoperable, thus reducing the effort necessary to establish relations with other LOs and create big courses. A visual tool for doing this shows also helpful, as it allows teachers to define courses easy and visually. Since our compilation creates standard PDDL files, we can use any planner that is currently available, which is clearly an important advantage. But our experiments have proved that CBP techniques are more interesting for adaptation when problems appear during execution, finding better quality and stabler solutions than traditional techniques.

Based on our experience, students seem more enthusiastic about personalisation in e-learning than teachers. Adopting this and other types of personalisation still raise further challenges and its horizon is not fully clear. The possibility of directly encoding in the e-learning standards all the information related to temporal, collaboration and resources constraints is still open, and it would increase the effective applicability of planning. And not only for planning application, but also for other approaches that address these issues.

**Acknowledgments.** This paper was partially funded by the Consolider AT project CSD2007-0022 INGENIO 2010 of the Spanish Ministry of Science and Innovation, the MICINN project TIN2011-27652-C03-01, the Valencian Prometeo project 2008/051 and the BW5053 research project of the Free University of Bozen-Bolzano.

## References


Brusilovsky, P., and Vassileva, J. 2003. Course sequencing techniques for large-scale web-based education. *International Journal Continuing Engineering Education and Lifelong Learning* 13(1/2):75–94.

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

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proc. 16th Int. Conference on Automated Planning and Scheduling (ICAPS-2006)*, 212–221. AAAI Press.

Garrido, A.; Onaindia, E.; Morales, L.; Castillo, L.; Fernandez, S.; and 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-ICAPS 2009)*, 18–27.

Garrido, A.; Fernandez, S.; Morales, L.; Onaindia, E.; Borrajo, D.; and Castillo, L. 2012. On the automatic compilation of e-learning models to planning. *The Knowledge Engineering Review* In press.

Garrido, A.; Onaindia, E.; and Sapena, O. 2008. Planning and scheduling in an e-learning environment. a constraint-programming-based approach. *Engineering Applications of Artificial Intelligence* 21(5):733–743.

Gerevini, A., and Serina, I. 1999. Fast planning through greedy action graphs. In *Proc. 16th Nat. Conference of the American Association for AI (AAAI-99)*, 503–510. AAAI Press.

Idris, N.; Yusof, N.; and Saad, P. 2009. Adaptive course sequencing for personalization of learning path using neural network. *Int. J. Advance. Soft Comput. Appl.* 1(1):49–61.

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(1-2):398–406.

Leake, D. 1996. *Case-Based Reasoning*. Cambridge, Massachusetts: The MIT Press.

Limongelli, C.; Sciarrone, F.; Temperini, M.; and Vaste, G. 2009. Adaptive learning with the LS-PLAN system: a field evaluation. *IEEE Transactions on Learning Technologies* 2(3):203–215.

LOM. 2002. Draft standard for learning object metadata. IEEE. rev. 16 february 2005. Available at http://ltsc.ieee.org/wg12/files/IEEE_1484_12_03_d8_submitted.pdf (accessed March 2012).

Martinez, F.; Magoulas, G.; Chen, S.; and Macredie, R. 2004. Recent soft computing approaches to user modeling in adaptive hypermedia. In *Proceedings of Adaptive Hypermedia (AH'04), LNCS 3137*, 104–114.

Polsani, P. 2003. Use and abuse of reusable learning objects. *Journal of Digital Information* 3(4). Available at http://journals.tdl.org/jodi/article/view/89/88 (accessed March 2012).

Roubickova, A., and Serina, I. 2012. Case-based merging techniques in oakplan. Technical report, Free University of Bozen-Bolzano, Italy.

SCORM. 2004. Sharable Content Object Reference Model. Available at http://http://www.adlnet.gov/capabilities/scorm (accessed March 2012).

Serina, I. 2010. Kernel functions for case-based planning. *Artificial Intelligence* 174:1369–1406.

Smyth, B., and McKenna, E. 2001. Competence models and the maintenance problem. *Computational Intelligence* 17(2):235–249.

Tonidandel, F., and Rillo, M. 2002. The FAR-OFF system: a heuristic search case-based planning. In *AIPS*, 302–311.

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

Yang, Q.; Nau, D.; and Hendler, J. 1992. Merging separately generated plans with restricted interactions. *Computational Intelligence* 8(4):648–676.