Chapter IX

AI Techniques for Monitoring Student Learning Process

David Camacho
Universidad Autonoma de Madrid, Spain

Álvaro Ortigosa
Universidad Autonoma de Madrid, Spain

Estrella Pulido
Universidad Autonoma de Madrid, Spain

María D. R-Moreno
Universidad de Alcaláj, Spain

ABSTRACT

The evolution of new information technologies has originated new possibilities to develop pedagogical methodologies that provide the necessary knowledge and skills in the higher education environment. These technologies are built around the use of Internet and other new technologies, such as virtual education, distance learning, and long-life learning. This chapter focuses on several traditional artificial intelligence (AI) techniques, such as automated planning and scheduling, and how they can be applied to pedagogical and educational environments. The chapter describes both the main issues related with AI techniques and e-learning technologies, and how long-life learning processes and problems can be represented and managed by using an AI-based approach.
INTRODUCTION

The e-learning (Clark, 2001; Kozma, 1991; Meyen et al., 2002) research field has become a hot topic in recent years. Many educators have seen it as a way to re-use previous courses stored in a database, or in other electronic formats (Schmitz, Staab, Studer, Stumme, & Tane, 2002), and to give flexibility to existing ones. Moreover, the increasing computing power and the available network infrastructure allows sharing and distributing these courses among public institutions and private corporations. These new educational approaches are evolving to use the new information technologies, and the Internet, as a virtual platform where all the involved people can implement new ways of communication.

Current e-learning techniques are modifying the traditional learning environment with a classroom, desktops with students, and a blackboard. These new techniques offer individualised contents and learning methodologies, which traditional courses cannot provide, and allow advanced learners to speed through or bypass contents that are redundant, whereas beginners slow down through them (Small & Lohrasbi, 2003). The progress made by each student can be monitored in order to determine the main problems that the students face when studying the units of a course. By knowing those problems, it is possible to propose e-learning activities that can improve the quality of the learning process and, as a consequence, improve the learning designs.

A learning design (LD) can be defined as an application of a pedagogical model for a specific learning objective, a target group, and a specific context or knowledge domain (Koper & Olivier, 2004). Different systems have been implemented to help course designers to specify and implement LDs. Two examples are the open-source system learning activity management system, or LAMS (LAMS, 2006), or the course management system Moodle (Moodle, 2006), which supports sequences of activities that can be both adaptive and collaborative. The different research works in the e-learning area led to the development of the IMS Learning design specification which is currently used as a standard format for learning designs (IMS LD, 2006). This specification is based on a metalanguage which allows modelling learning processes. In IMS LD model concepts like roles, activities, or environments are defined for describing learning designs.

In higher education, the increasing tendency is to create virtual learning environments (VLE) which are designed to facilitate teachers the management tasks of educational courses for their students. This increasing number of platforms, systems and tools related to virtual education has led to the creation of different e-learning standards. These standards, such as SCORM (2006), have been developed to facilitate the utilization (and reutilization) of teaching materials (through the definition and creation of learning objects). Currently, these technologies and standards are mature enough to incorporate innovative techniques that could provide new mechanisms to deal with learning processes.

The new virtual learning environments provide an interesting field for different kinds of researchers. We will focus on artificial intelligence (AI) researchers that can experiment with their automatic problem solving algorithms, or develop and design new algorithms in this complex domain; and educational researchers that can use a new kind of tools and techniques that could aid to detect, reason, and solve (automatically) deficiencies detected in their initial learning designs. One of the areas of AI most suitable to be applied within this context is the automated planning and scheduling. Planning techniques generate a plan (sequence or parallelization of activities) that achieves a set of goals given an initial state and satisfies a set of domain constraints represented by operators schemas. In scheduling systems, activities are organised along the time line by having in mind the resources available. These systems can perfectly handle temporal reasoning.
and resource consumption, together with some quality criteria (usually centred around time or resource consumption) but they cannot produce the required activities and their precedence relations given that they lack an expressive language to represent activities. These techniques have been applied with success in different real (and complex) environments such as industry, robotics, or information retrieval. Of special interest in the last few years has been the development of autonomous architectures (Muscettola, Dorais, Fry, Levinson, & Plaunt, 2002) that can carry out a large number of functions, such as tracking a spacecraft’s internal hardware or rover’s position, ensuring the correct working, and repairing when possible, without (or little) human intervention. In these new operation models, scientists and engineers communicate the high-level goals to the spacecraft or to the rovers, which are translated into planning and/or scheduling sequences. Then, a continuous status checking is performed in order to detect any damage, and, finally, the plan is executed. These systems must also have the capability to understand that the errors occurred during the process of accomplishing the goals. This chapter shows how this kind of AI-based techniques can be appropriately used into e-learning, and more specifically into virtual education or VLE domains (Sicilia, Sánchez-Alonso, & García-Barriocanal, 2006). We will apply these techniques to a specific e-learning tool called Task-Vased Adaptive Learner Guidance on the Web (TANGOW) developed by some of the authors of this chapter (Carro, Pulido, & Rodriguez, 1999b).

**REPRESENTATION FORMALISMS IN LEARNING DOMAINS**

In this section we will describe different formalisms that have been used in e-learning systems to represent (a) the learning area (domain model) which includes the course concepts and the relationships between them, and (b) the current situation of a given learner with respect to the whole learning process. These models will be later considered, by using a particular e-learning tool, to understand how traditional AI techniques can be incorporated into a particular e-learning system.

Several standards and guides have been proposed related to learning object metadata, student profiles, course sequencing, and so forth. The IEEE Learning Technology Standards Committee (LTSC, 2006) has developed the learning object metadata (LOM, 2006) standard which specifies the attributes required to describe a learning object, where a learning object is defined as any entity, digital or nondigital, which can be used, re-used or referenced during technology supported learning. Relevant attributes of learning objects to be described include type of object, author, owner, terms of distribution, format, and pedagogical attributes, such as teaching or interaction style. The standard also defines how LOM records should be represented in XML and RDF.

Promoting Multimedia Access to Education and Training in European Society (PROMETEUS) tries to apply the IEEE LTSC standards into Europe context and cultures.

Another specification which allows the modeling of learning processes is the learning design (LD) information model (IMS LD, 2006) from the IMS Global Learning consortium. A learning design is a description of a method enabling learners to attain certain learning objectives by performing certain learning activities in a certain order in the context of a certain learning environment.

LD is designed to integrate with other existing specifications. Among these, it is worth mentioning the IMS content packaging (IMS CP, 2006), which can be used to describe a learning unit. A learning unit can have prerequisites which specify the overall entry requirements for learners to follow that unit. In addition, a learning unit can have different components such as roles and activities.
Roles allow the type of participant in a unit of learning to be specified. There are two basic role types: learner and staff. Activities describe the actions a role has to undertake within a specified environment composed of learning objects.

LD also integrates the IMS simple sequencing (IMS SS, 2006), which can be used to sequence the resources within a learning object as well as the different learning objects and services within an environment. Content is organized into a hierarchical structure where each activity may include one or more child activities. Each activity has an associated set of sequencing rules which describe how the activity or how the children of the activity are used to create the desired learning experience. The learning process can be described as the process of traversing the activity tree, applying the sequencing rules, to determine the activities to deliver to the learner.

The general format of a sequencing rule can be expressed informally as: if condition set then action. There may be multiple conditions. Conditions may be combined with a single and combination (all conditions must be true) or a single or combination (only one condition must be true). Individual condition values may be negated before being combined in the rule evaluation.

The U.S. Federal Government Advanced Distributed Learning (ADL) initiative has also proposed a model called shareable courseware object reference model (SCORM) which describes how the Department of Defense will use learning technologies to build, and use the learning environment of the future. The standard defines what is called “Learning Object Metadata,” which is a dictionary of tags that are used to described learning content in a variety of ways. For a given learning object, these metadata describe, for example, what the content is, who owns it, what costs (if any), technical requirements, educational purpose, and so forth. The order in which learning objects are presented to the learner is specified by using sequencing rules. A sequencing rule has the format if condition then action, where condition can be chosen among, for example, “completed,” “score less than,” or “time limit exceeded.” On the other hand, an action could be, for example, “skip,” “disable,” or “hide from choice.”

The Aviation Industry CBT (Computer-Based Training) Committee (AICC) is in charge of developing guidelines for the aviation industry in the development, delivery, and evaluation of CBT and related training technologies.

In addition to these standards, there are other specific proposals such as the GRAFCET representation formalism described in M’tir, Jeribi, Rumpler, and Ghazala (2004), which uses a graph to represent the sequences of course concepts and the possible learning itineraries.

Other existing work (Ahmad, Basir, & Hassanein, 2004) uses fuzzy logic to relate attributes in the learner module and concepts in the domain model. The motivation for the use of fuzzy logic is that it is appropriate for representing and reasoning with vague concepts and that the formalisation of the level of understanding of a given concept by a learner is an inherently vague process.

**AUTOMATED PROCESSES IN E-LEARNING TOOLS**

Most of the current VLE contain prefixed courses where the user navigates and learns the concepts that they have been planned for. Some e-learning tools include situation learning (SL) courses where the user is presented with different predefined situations where the user has to choose among different options. The drawback of this type of course is that nothing is dynamically generated and a lot of effort is required to create challenging situations that keep the user’s attention.

Although the instructors can get statistics as well as other information about the student progress, there is still a lack of feedback among the previous users, the tool, the instructors, what the user is interesting in, and the future users. Among the tools that have worked on this direc-
AI Techniques for Monitoring Student Learning Process

In addition to the CourseVis system (Mazza & Dimitrova, 2003) and the dynamic assembly engine (Farrell, Liburd, & Thomas, 2004), an approach for automatic course generation (in some ways similar to the one presented in this chapter) is the work of Ulrich (Ullrich, 2005) who uses an AI hierarchical task network (HTN) planner called JSHOP (Ilghami & Nau, 2003) which assembles learning objects retrieved from one or several repositories to create a whole course. Our approach not only can link learning objects, but also schedule them along a period of time and consider previous student results to generate different learning designs.

Since our goal is to monitor the learning process in TANGOW, the next subsections present a review of the main existing techniques: AI Planning and Scheduling.

INTRODUCTION TO PLANNING TECHNIQUES

Planning can be defined as the sequence or parallelization of activities that, given an initial state, achieves a set of goals and satisfies a set of domain constraints represented as operators schemas. Using a high level description, the inputs of these systems are shown in Figure 1:

- **Domain theory**: The STRIPS (Fikes & Nilsson, 1971) representation is one of the most widely used alternatives. A world state is represented by a set of logical formulae, the conjunction of which is intended to describe the given state. Actions are represented by the so-called operators. An operator consists of **preconditions** (conditions that must be true to allow the action execution), and **post-conditions** or effects (usually consisting of an add list and a delete list). The **add list** specifies the set of formulae that are true in the resulting state, while the **delete list** specifies the set of formulae that are no longer true and must be deleted from the description of the state. A course can be defined in terms of a set of learning activities that are performed by students.

- **Problem**: Described in terms of an initial state and a goal. The initial state is represented by logical formulae that specify the situation for which a solution is being looked for. Examples of initial states in a learning environment would be the students previous knowledge, the resources that a course uses and the time period when they are available, and so forth. Goals are often viewed as specifications for a plan. In a learning environment, a possible goal would be that the student is able to apply critical thinking to a specific subject.
Some AI planners include a third input referred to as control knowledge. It could guide the solver to the right alternatives of the search tree potentially avoiding backtracking and arriving straight forward to the solution.

As an output, planners generate a plan with the set of operators that achieves a state (from the initial state) that satisfies the goals.

The main AI planning techniques are described next:

- A **total order (TO) planner** generates solutions that are sequences of total ordered actions. The basic structure is a tree where nodes can be plans or states, and edges are actions or state transactions; then any search algorithm can be applied.

- In a Partial Order planner, nodes represent partially specified plans, and edges denote plan-refinements operations such as the addition of an action to a plan. The planning algorithm commits to only the essential ordering decisions. There is no need to prematurely commit to a complete, total sequence of actions.

- A **Graphplan planner** alternates between graph expansion and solution extraction. The graph expansion extends the plan graphs forward until it has achieved a necessary condition for plan existence. The solution extraction phase performs a backward-chaining search on the graph, looking for a plan that solves the problem. If no solution can be found, the cycle repeats the expansion of the planning graph.

- A **heuristic search planner (HSP)** transforms planning problems into problems of heuristic search by automatically extracting heuristics functions from STRIPS encoding instead of introducing them manually. The bottleneck is the computation in every new state of the heuristic from scratch.

- An **SAT-based planner** takes a planning problem as an input, guesses a plan length, and generates a set of propositional clauses that are checked for satisfiability. After the translation is performed, fast simplification algorithms are used to solve the problem.

- An **HTN planner** uses tasks networks and tasks decomposition (methods). A task network is a collection of tasks that need to be carried out, together with constraints on the order in which tasks can be performed. The basic algorithm is to expand tasks and resolve conflicts iteratively, until a conflict-free plan can be found that consists only of primitive tasks.

- A **Markov decision process (MDP)** is defined by an initial distribution over the states, the dynamics of the system with states annotated by different possible actions, the probabilistic state transitions, and a reward function to make the transition from one state to another. This kind of techniques requires full enumeration of all possible states what can make it intractable in most of the planning systems. Most of the work in this area has focused on using only a subset (the most probable state space) or abstractions of the state space.

- A **contingent plan** refers to a plan that contains actions that may or may not actually be executed, depending on the circumstances that hold at the time. Another way to handle uncertainty is by applying probabilistic planners which use probabilities of the possible uncertain outcomes to construct plans that are likely to succeed.

**INTRODUCTION TO SCHEDULING TECHNIQUES**

Scheduling can be defined as the process of organising activities along the time line by taking into account the resources available. Many
AI Techniques for Monitoring Student Learning Process

Techniques used in the area of scheduling systems come from the operational research (OR) area (i.e., branch and bound, simulated annealing, Lagrangian relaxation). Lately, constraint programming (CP) has been applied to different scheduling problems with very good results, that is, job-shop scheduling and the RCPSPmax problem (Kolisch & Hartmann, 1999). A RCPSPmax consists of a set of activities where two kinds of constraints can be interrelated:

- **Precedence constraints** that impose the restriction that an activity cannot start before its predecessor activities, and
- **Resource constraints** among activities that consume the same resource due to the limited capacity of the resource itself.

The objective is to find precedence and resource assignments for all the activities in the horizon imposed. Figure 2 shows a simple example of a Job-Shop Scheduling problem with two resources: resource R1 with a capacity of 2, and resource R2 with a capacity of 3. The left part of the figure shows the precedence constraints among activities and the resources that each one requires. The right part of the figure shows the solution to the problem. Since R1 has a maximum capacity of 2, the 3 activities that consume this resource cannot be performed in parallel. Then, the scheduler will add a precedence constraint to one of them. Resource R2 has a capacity of 3 but none of the activities that require this resource have to be executed in parallel, so there are no conflicts. If we impose a deadline of 5 time units to the original problem (we consider that each activity has a duration equal to 1), the solution given by the scheduler will be also time consistent. However, a value lower than 5 will make the solution inconsistent.

Then, scheduling techniques can be easily generalized and applied to a learning environment. In this case, instead of having machines and jobs (Job-Shop Scheduling problem), we have students, educators, and learning units (LU) in courses.

Each learning unit (operation) needs to be processed during a period of time for a given student (machine), and the unit will be supervised by an educator. The course will also have a limited duration (deadline). Each instructor will have a maximum number of students (we consider an instructor as a resource with a total resource capacity given by the number of students the instructor is able to advise). We need to know the initial and end time of each LU considering precedence constraints among them. The variable values are imposed by the problem conditions: learning activity durations, course duration, number of learners, and so forth. AI integrated planner and scheduler systems generate a plan or a set of plans if a solution exists for the given deadline. A plan can be seen as a sequence of operator applications (learning activities) with a specific duration that can lead from the initial state to a state in which the goals are reached with the resources available.

**Figure 2.** A job-shop scheduling example with two resources
In a specific learning design, we need to impose a deadline, that is, the total duration of the course, and the resources that are available, that is, the number of educators. Then, it is up to the educator and the pedagogical responsible to study the best way to distribute the number of hours and their contents among the different units in order to assure the quality of the education process. This task can be done automatically by applying Planning and Scheduling techniques to a new domain: e-learning environments. Although the process will be explained in detail in the following sections, the basic idea is to change some parameters and to use the feedback from the students that have already followed the course.

**TANGOW: A TOOL FOR VIRTUAL EDUCATION**

TANGOW facilitates the development and deployment of adaptive courses. In these courses the contents, the navigational options, and the flexibility of the guidance process are adapted to both the user features and their actions while interacting with the course. Adaptivity is an important feature because the lifelong learning philosophy is growing in importance in many environments and the same virtual education course can be accessed by students with different backgrounds, age, and interests.

Teachers can describe adaptive courses by means of tasks and rules. *Tasks* are the basic units in the learning process. They include topics to be learned, exercises to be done, examples to be observed, and so forth, that is, tasks to be performed in order to learn or put into practice the concepts or procedures involved within the course. *Rules* specify the way of organizing tasks in the course along with information about the task execution (order among tasks—if any—free task selection, prerequisites among tasks, etc.).

Figure 3 presents an example of the (partial) structure of a TANGOW course on operative systems, as taught to second year students of a computer science degree. The current version of the course is composed by a number of tasks representing *theory* units and also *examples*.

The example shows how tasks are decomposed into subtasks according to specific rules. The complete course consists of four tasks: *operative system overview*, *operative system concepts*, *distributed systems* and *security*. These tasks are combined through an AND rule, which states that all the subtasks must be completed exactly in the order in which they are listed in the rule. The *operative system overview* task is divided into *services*, *security*, and *architecture* subtasks, combined through an OR rule. The OR rule dictates that, in order to complete the composed task, at least one of the subtasks must be completed, in any chronological sequence. Differently, the *operative system concepts* task is decomposed, through an ANY rule, into the *process*, *memory*, *scheduling*, *input-output*, and *file* subtasks. An ANY rule means that all the subtasks must be completed but the order is not relevant. Following with the course, it can be seen that, in order to execute the *Scheduling* task, the student has to sequentially execute the *scheduling principles* and *scheduling algorithms* tasks. *Scheduling algorithms* consists of the study of five specific algorithms in any order. Finally, the task *round-robin* algorithm can be learned by reading either the *description* or any of the three *examples* provided.

Besides the correct sequencing, the rule may state conditions that must be fulfilled in order for the rule to be applied. These conditions are expressed by means of attribute values regarding user features, such as personal characteristics (age, language, experience, etc.), learning style (visual/verbal, intuitive/deductive, etc.) (Paredes & Rodriguez, 2002), preferences (type of information desired, learning strategy, etc.), and actions while interacting with the course (tasks visited, exercises performed, results obtained in the tests, time spent in every task, and so on). The latter type of attributes are called “dynamic attributes”
AI Techniques for Monitoring Student Learning Process

Because rules can depend on dynamic attributes, the decomposition of the next task can only be computed just on time when the student selects the given task. In this way, the TANGOW system consults the course description and the data about the student and generates, step by step, a personalized course for each student, adapting the different course aspects to each student.

**Authoring a TANGOW Course**

When designing an adaptive course for TANGOW, the first step is to establish the user features to be considered for the adaptation. The attributes selected to be used in rule conditions compose the user model. These data are stored in the student database along with the log files containing the sequence of actions performed by the students. For example, regarding the *operative system* course...
AI Techniques for Monitoring Student Learning Process

Figure 4. Task decomposition based on student characteristics

![Task decomposition diagram]

The course analyzed in Figure 4, the relevant student feature is the student role (end user, application programmer, OS designer). The list of visited tasks will also be needed, as there are prerequisite relationships between some of them (for example, processes is required for scheduling algorithms, Figure 3).

Afterwards, the designer describes the adaptive course itself (Carro, Ortigosa, & Schlichter, 2003; Carro, Pulido, & Rodriguez, 1999a) by specifying the tasks and rules that will be part of the course, as well as the contents (generally HTML files) associated with each task and used for page generation. The designer can specify different variations of any of these aspects in order to adapt the course to student features and actions.

Each task will be defined as atomic or composed. Atomic tasks will be the leaves of the task tree, while composed tasks are the inner nodes and will have one or more rules describing how it is decomposed into subtasks. Table 1 shows some rules describing how composed tasks should be divided into subtasks.

**TANGOW Logs**

While the student is interacting with the course, all of the student’s actions are logged. This log stores information about the tasks the student has

<table>
<thead>
<tr>
<th>Task name</th>
<th>Conditions</th>
<th>Subtasks</th>
<th>Sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>‘Operative System Overview’</td>
<td>role = ‘Application programmer’</td>
<td>‘Services’, ‘Security Overview’</td>
<td>ANY</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
AI Techniques for Monitoring Student Learning Process

Figure 5. A portion of a TANGOW log file

```
<log-root>
  <user-model>
    ...
  </user-model>
  <log-activity>
    <log activity="Operative System Overview" action="START-ACTIVITY" timestamp="2006-11-26T09:52:00"/>
    <log activity="Services" action="START-ACTIVITY" timestamp="2006-11-26T09:54:15"/>
    <log activity="Services" score="0.5" action="END-ACTIVITY" timestamp="2006-11-26T13:02:55"/>
    ...
  </log-activity>
</log-root>
```

visited, the corresponding time stamps, the level of completeness, and the score obtained, when it applies. Figure 5 displays a partial example of a student interacting with the adaptive course.

Log files enable the course designer to retrace the interaction of the student with the system/course, and can be used with different goals. For example, Ortigosa and Carro (2003) use the information contained in the log files to provide the course designer with information about possible problems or improvement opportunities within the adaptive course.

MONITORING TANGOW COURSES: A CASE STUDY

The e-learning methodology proposed uses automated reasoning techniques, such as planning and scheduling, to automatically learn from possible mistakes in the learning design process and to look for new solutions (Camacho & R-Moreno, 2007; R-Moreno & Camacho, 2007). This methodology has been implemented in a planning and scheduling (P/S) system called IPSS (R-Moreno, Oddi, Borrajo, & Cesta, 2006). In this section, details will be presented about how the IPSS and the TANGOW systems have been integrated and the advantages obtained with this integration.

TANGOW & IPSS Integration

In IPSS (R-Moreno et al., 2006), reasoning is divided into two levels. The planner module (IPSS-P) focuses on the action selection, and the scheduler module (IPSS-S) on the time and resource assignment. Figure 6 shows in more detail how the different modules (layers) of IPSS interact. Since our planner is a total order planner (the solution is a sequence of activities), it is not enough to look for a solution that minimises the time and the resources. We use a de-order algo-
AI Techniques for Monitoring Student Learning Process

Algorithm (Bacstrom, 92) to eliminate the unnecessary causal links. Then, the solution generated by the de-order algorithm is given to the temporal and resource reasoners. During the search process, every time the planner chooses an operator, it consults the scheduler for the time and resource consistency. If the resource-time reasoner finds the plan inconsistent, then the planner backtracks. If not, the operator is applied, and the search process continues. To know more details about the algorithm, readers can refer to (R-Moreno, 2003; R-Moreno et al., 2006).

Figure 7 shows the architecture of the system that results from the integration of the IPSS and the TANGOW systems. The monitored learning process can be described as follows:

1. The educators define:
   a. The teaching tasks and rules to build the adaptive course (by using the TANGOW tool).
   b. The educator assigns to each task both a priority and a time estimation associated to the task (a high priority means that the task is considered as essential in the learning process).
   c. The teacher selects the number and kind of dimensions that will be used to generate the metadata. In our example, the teacher has selected the knowledge-student level (end user, application programmer, OS Designer).

2. The students interact with TANGOW. These interactions generate different logs that will be stored in the system.

3. By using the above-mentioned information (students logs, teaching tasks, rules, task priorities, time estimation) the metadata is generated based on both logs and educator estimations.

4. The metadata is appropriately mapped into an IPSS representation. This mapping process generates the domain and the initial problem that will be used by IPSS to solve the defined problem.

5. IPSS looks for solutions that solve possible problems existing in the initial learning design by taking into account the teaching

Figure 7. IPSS integration for a specific TANGOW course

![TANGOW Integration Diagram]
task decomposition, their priorities and estimated duration time.

6. For each kind of student (end user, application programmer, OS designer) a plan is generated with a possible scheduled course.

The following subsections describe the previously listed processes in detail by using the TANGOW course example of Figure 3.

## From TANGOW Metadata to P/S-Based Problems

As mentioned, the planner domain theory contains all the actions represented by operators. The language for describing an IPSS domain theory is based on an augmentation of the representation originally proposed by Fikes and Nilsson (Fikes & Nilsson, 1971). Since this representation is quite restrictive, it has been extended to allow disjunctive preconditions, conditional effects and universally-quantified preconditions and effects, quality metrics, durations, time and resource constraints, and continuous values.

The first step for defining a domain consists of identifying the operators and the object types that are needed in the domain (for declaring the type of each operator variable). Types can be defined as structured in a hierarchy. A special type, the infinite type, allows representing continuous valued variables, while finite standard types represent nominal types. In our domain, we have, among others, the following types: STUDENT, ROLE, SUBTASK, COURSE, DURATION, PRIORITY, and TIME. Variables of type STUDENT instantiate to the possible student stereotypes. The variable ROLE represents the relevant user features (end user, application programmer, OS designer) and COURSE represents the courses that we want to track. Finally, DURATION, PRIORITY, and TIME are infinite types that allow us to handle numerical values needed to calculate the duration and priorities of each task. By following the TANGOW example described in Section 4 (Figures 3, 4, and 5 and Table 1), we can obtain the needed metadata for IPSS (see Table 2).

The IPSS operator in Figure 8 is composed of the following fields:

### Table 2. TANGOW metadata

<table>
<thead>
<tr>
<th>Task name</th>
<th>Conditions</th>
<th>Subtasks</th>
<th>Sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Task</td>
<td>Priority</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>‘Operative System Overview’</td>
<td>role = ‘End user’</td>
<td>‘Services’</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘Security Overview’</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘Architecture Overview’</td>
<td>3</td>
</tr>
<tr>
<td>‘Operative System Overview’</td>
<td>role = ‘Application programmer’</td>
<td>‘Services’</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘Security Overview’</td>
<td>2</td>
</tr>
<tr>
<td>‘Operative System Overview’</td>
<td>role = ‘OS Designer’</td>
<td>‘Services’</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘Security Overview’</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘Architecture’</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
AI Techniques for Monitoring Student Learning Process

- **Params field**: Contains a list of the variables whose values will be printed out through the user interface when a solution is found.
- **Preconds field**: Contains the preconditions of the operator.
- **Effects field**: Contains the add effects of the operator.
- **Constraints field**: Contains the temporal constraints.

The symbols within < > are variables that are instantiated during the problem solving process. The operator in Figure 8 has two preconditions:

\[(\text{compose-subtask } \text{End User} <\text{tk}>, <\text{tk1}>, <\text{tk2}>)\]
and \[(\text{student-role } <\text{st}>, <\text{role}>)\] and three add effects:

\[(\text{done } <\text{st}>, <\text{tk}>)\], \[(\text{done } <\text{st}>, <\text{tk1}>)\] and \[(\text{done } <\text{st}>, <\text{tk2}>)\].

This operator, called “T_EU_Operating-System_Overview,” corresponds to the “Operating System Overview” task under the “End User role” condition in Table 2 which is composed of “Services,” “Security Overview,” and “Architecture Overview” substasks. These subtasks are represented within the operator as the variables \(<\text{tk}>, <\text{tk1}>, <\text{tk2}>\). The \(<\text{role}>) variable will be instantiated by the End_user value.

The variables <pt>, <pt1>, and <pt2> represent priorities and can take numbers as values. We need to use the **gen-from-pred** IPSS function to constraint the values that these three variables can take. This function generates a list of values as the bindings for a variable by using the information on the current state. In this example, in the case of <pt>, **gen-from-pred** returns the list of values \{x\} greater or equal to 1 such that the

\[
(\text{OPERATOR T_EU_OperatingSystem_Overview})
\]

\[\text{(params } <\text{st}>, <\text{tk}>, <\text{tk1}>, <\text{tk2}>\)\]
\[\text{(preconds}\]
\[\text{((<st> STUDENT))}\]
\[\text{(<?role> ROLE))}\]
\[\text{(<?tk> SUBTASK))}\]
\[\text{(<?tk1>(and SUBTASK(diff <tk1>))))}\]
\[\text{(<?tk2>(and SUBTASK(diff <tk1> <tk2>)))}\]
\[\text{(<?pt>(and PRIORITY (gen-from-pred (prioritySe <role> <tk> <pt>))\]
\[\text{(>= <pt> 1))}\]
\[\text{(<?pt1>(and PRIORITY (gen-from-pred (prioritySO <role> <tk1> <pt1>))))}\]
\[\text{(>= <pt1> 1))}\]
\[\text{(<?pt2>(and PRIORITY (gen-from-pred (priorityAO <role> <tk2> <pt2>))))}\]
\[\text{(>= <pt2> 1))}\]
\[\text{(and)}\]
\[\text{(compose-subtask End_User <tk> <tk1> <tk2>)}\]
\[\text{(student_role <st> <role>))}\]
\[\text{(effects}\]
\[\text{()}\]
\[\text{(add (done <st> <tk>))}\]
\[\text{(add (done <st> <tk1>))}\]
\[\text{(add (done <st> <tk2>))}\]
\[\text{(constraints}\]
\[\text{(<?dur> (and DURATION constraint-from-pred (durationSe <role> <tk> <dur>))))}\]
\[\text{(<?dur1> (and DURATION constraint-from-pred (durationSO <role> <tk1> <dur1>))))}\]
\[\text{(<?dur2> (and DURATION constraint-from-pred (durationAO <role> <tk2> <dur2>))))}\]
\[\text{(<?time> (and DURATION calculate-total-duration <dur> <role> <dur2>))))}\]
\[\text{(TIME <time>))}\]

Figure 8. An IPSS operator corresponding to the “operating system overview” task
literal (prioritySeqEnd_user Services x) is true in the current state.

The variables \(<\text{dur}>, <\text{dur}1>, \text{and } <\text{dur}2>\) represent task duration. As the values they can take are lists of two elements corresponding to the minimum and the maximum duration, we need to use the \textit{constraint-from-pred} IPSS function to constraint the values that these three variables can take. This function generates a list of values to be possible bindings for the corresponding variable by using the information of the current state referred to the duration of each task that compose a specific task. IPSS will choose a possible integer value in that range during the problem resolution.

In order to implement static properties efficiently, IPSS allows user-defined functions to represent them. In our example, we have defined the function \textit{diff}, that allows us to calculate if the objects passed as arguments are different; and the function \textit{calculate-total-duration}, that permits encoding the function: \textit{time}= \textit{dur}+\textit{dur}1+\textit{dur}2.

The second input to the planner is the problem to be solved, described in terms of an initial state and a set of goals to be achieved. The description of an initial state is composed of a list of objects and their corresponding types together with a set of instantiated predicates (i.e., literals) that describe the configuration of those objects. The objects in the state must be instances of the types that are declared in the domain. Figure 9 shows some initial conditions and two goals corresponding to two student stereotypes. The goals are that learner1 whose student role is \textit{end user} and that learner2 whose student role is \textit{application programmer} must learn the Operative System course.

Monitoring TANGOW Courses

Once the metadata is generated, there are two different planning/scheduling possibilities. On one hand, it could be the first time that this course is executed. In this situation, the planner does not have information about how much time is necessary for a particular task (we use the educator time estimation). The system will propose a schedule that can be dynamically modified once the interactions with the students provide the initial time durations for the tasks (a teaching task that requires more than the scheduled time will force to modify the subsequent tasks, see Figure 10).

Figure 10 shows the execution of different teaching tasks proposed by the planner. Any task could be under or over estimated depending on external factors (students skills, labs/classrooms availability, and so forth). Although the total duration (i.e., makespan) of the course is fixed, IPSS can replan the remaining tasks in order to fit them in the remaining time. IPSS will either increment/decrement the duration of the tasks based on the priorities, or add/delete subtasks based on the AND, OR, and ANY rules. Figure 9 shows a possible course execution scenario, in which Task1 was under estimated by the educators (when accessing the course, it took longer to students to execute the task). The new time duration (including a time increment) will be managed by the planner/scheduler in the next cycle to adjust the time duration of the rest of tasks that have not been executed yet. The planner/scheduler main objective is to execute all the teaching tasks in the available time. For this reason, over estimations (i.e., Task 2) and under estimations (i.e., Task 1) will be sequentially used in every planning/scheduling cycle to (dynamically) adapt the available time. This adaptation could result in the addition or removal of low priority teaching tasks.

On the other hand, if a particular course has been executed several times, we can access the student logs to look for a \(<\text{min, max}>\) estimation for the time duration of a particular task. If the majority of students a task requires less time than estimated by the educator, this means that the next time the students start the course we can consider that duration as the baseline duration. This information will be given to IPSS that will adjust the whole course based on that change. In this way, IPSS will provide a better initial
AI Techniques for Monitoring Student Learning Process

Figure 9. An example of IPSS initial conditions

(objects
(learner1 learner2 STUDENT)
(End_user Application_programmer OS_Designer ROLE)
(Services Security_Overview Architecture_Overview SUBTASK)
(Operating_System COURSE))

(state
and
; SubTasks that compose the tasks depending on the student role
(compose-subtask End_user Services Security_Overview Architecture_Overview)
(compose-subtask Application_programmer Services Security_Overview)
; . . .
; Features for each student stereotype
(student_role learner1 End_user)
(student_role learner2 Application_programmer)
; Priorities for each substask
(prioritySe End_user Services 1)
(prioritySO End_user Security_Overview 2)
(priorityAO End_user Architecture_Overview 3)

(prioritySe Application_programmer Services 1)
(prioritySO Application_programmer Security_Overview 2)
; ...
; Durations for each substask
(durationSe End_user Services (2 4))
(durationSO End_user Security_Overview (3 6))
(durationAO End_user Architecture_Overview (5 7)))

(durationSe Application_programmer Services (2 3))
(durationSO Application_programmer Security_Overview (2 4))

(durationSe OS_Designer Services (2 3))
; ...
(goal
(and (Learn learner1 Operative_System)
(Learn learner2 Operative_System)))

Figure 10. Sequential teaching tasks execution
solution in the next course execution. The same will occur if the time assigned to a task by the educator is underestimated. From the log files we can extract that information and translate it into IPSS to reschedule the whole course. If a rescheduling is needed, two things can happen: all the tasks can still be fit into the total time assigned to the course or they cannot. In the latter case, IPSS will have to eliminate one or more tasks from the course.

In both situations, with and without previous students information, IPSS can use the available information (in the first case a worse time assumption will be assumed) to schedule tasks. Figure 10 shows how the cyclic execution of a particular course can be used by IPSS to iterative improve the quality of the plans. A proposed schedule will have a higher quality if it is better adapted to both the students characteristics and the available time.

The IPSS Final Solution

Finally, IPSS generates a plan with the sequence of operators that achieves a state (from the initial state) that satisfies the goals and their start and end times. Figure 12 shows the solution generated from the initial conditions of Figure 9. In the solution, IPSS instantiates each operator (that is, each task in TANGOW) by giving value to the task starting and ending times, and to the subtask list that compose the task. The fact that some tasks (operators instantiated) have prerequisite relationships with others already performed, imposes the restriction that the starting time of a task must be later in time to the ending time of its prerequisite tasks. This is the case of the “T_EU_OperatingSystemConcept” and “T_EU_OperatingSystemOverview” tasks in Figure 11. The starting time of “T_EU_OperatingSystemConcept” is equal to the ending time of “T_EU_OperatingSystemOverview”. These tasks belong to the tasks that have to be performed by the student with the “End User” role (see Table 2). Since the Sequencing is an “AND,” all the tasks must be performed sequentially. But these prerequisite relationships may not exist between other tasks such as the “T_AP_OperatingSystemOverview” and “T_EU_OperatingSystemOverview” tasks that can be executed in parallel. The “T_AP_OperatingSystemOverview” task will be performed by a student with the “application programmer” role that has no conflicts with the tasks performed by the “end user” student.
CONCLUSION

In the first e-learning systems, any student, no matter what the student’s personal features were, was presented with the same materials, the same exercises were proposed to him and in the same order. The next generation of e-learning environments are those that are able to adapt the deployed course to the features and actions of the student. In this way, every student will follow an individualised course specifically deployed for the student. We can refer to this kind of adaptation as an individual adaptation which is performed based on the individual characteristics and actions of each student accessing the system.

This chapter shows how it is possible to go one step further in the development of e-learning systems and implement a group-based adaptation based on the actions not of an individual student but of a set of students who have accessed the system along a period of time. The basic idea is to register student actions when interacting with an e-learning course based on an initial learning design. A Web-based learning system called TANGOW is used for this purpose. Then, planning and scheduling techniques as implemented in the IPSS system are applied to the data collected in the TANGOW log files in order to refine the initial learning design by adjusting the duration and the ordering of the activities proposed to the students accessing the e-learning system. For the integration, the TANGOW rules and conditions are translated into IPSS operators, and the TANGOW attributes into IPSS types to produce a plan that corresponds to a course instance (tasks dependency).

Consequently, the deployed course adapts itself not only to the personal features and actions of every student who accesses the course, but also to the global actions of a group of students. With this approach the courses generated are automatically validated avoiding inconsistencies in linking activities, durations given to each activity and the total duration of the course, saving time to the educators. Since the planning and scheduling techniques can be applied to the collected

<table>
<thead>
<tr>
<th>Operator-Name</th>
<th>Start-Time</th>
<th>End-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_EU_OperatingSystemOverview learner1 Services Services_Observation Architecture_Observation)</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>(T_EU_OperatingSystemConcept learner1 Processes Memory Scheduling Files)</td>
<td>10 28</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_EU_Security learner1)</td>
<td>66 70</td>
<td></td>
</tr>
<tr>
<td>(T_AP_OperatingSystemOverview learner2 Services Services_Observation)</td>
<td>0 4</td>
<td></td>
</tr>
<tr>
<td>(T_AP_OperatingSystemConcept learner2 Processes Memory Scheduling Input_Files)</td>
<td>4 30</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_AP_Security learner2)</td>
<td>57 60</td>
<td></td>
</tr>
</tbody>
</table>

#<THIS result: 10.34 secs, 70 this-makespan>
AI Techniques for Monitoring Student Learning Process

log files as many times as required, the process of improving and refining becomes a long-life process that will only stop when the designer or tutor considers it suitable.

Finally, we also want to mention the issues that AI planners can gain with this approach. Generally, to specify the domain theory, a deep understanding of the way AI planners work and its terminology is needed. However, if we use a tool like TANGOW, the description language is closer to the user and allows an automatic verification of the syntax through a friendly interface. Also, a new domain to apply and develop new AI algorithms has being created.

We are currently on the process of testing the proposed approach. In a first stage we are working with synthetic logs. The next stage will involve empirical tests with real users to verify improvements on the course.

FUTURE RESEARCH DIRECTIONS

The current state in computer-based education technologies, tools, and standards provides some new interesting perspectives to other research areas like artificial intelligence. The well established standards, such as IMS, SCORM, or LOM, are currently being used to define and develop new adaptive virtual-based education tools. These tools support the creation of personalized learning designs (LD). With these new designs it is possible to reuse and exchange useful information among different platforms. These new tools can be used by educators (and/or learning designers) not only to define the contents of the course (i.e., using the IMS LD specification), but also to create adaptable and personalized learning flows, so that the educational system can monitor and control the whole learning process. This chapter has described a particular authoring tool (TANGOW) that can be used to achieve the previous goal based on teaching rules, and has shown how a particular AI system that integrates planning and scheduling techniques can be used to improve the learning quality of a particular course. In our approach the term quality is used to describe the fact that suggesting one or several modifications, is a simple way to control and monitor a particular course. In this way, educators can easily detect hidden problems and improve, and reach, their final goals.

However, in the near future a new kind of adaptive intelligent education-based tools will be completely designed and developed using this kind of technique (other related AI-based techniques such as machine learning are currently used to learn students profiles and take learning decisions). In these new tools it will be possible to control, monitor, and automatically solve, several detected problems in the learning designs deployed. These problems could be automatically detected (by logging the interactions with the students, and analysing the quantitative results from tests and exams), or provided directly by educators and learning designers, as we have shown previously. These new tools will be able to automatically evaluate, through the long-life learning process of a particular course, the problems and automatically modify the learning designs in order to smooth them.

To achieve the previous goal it will be necessary to adapt and integrate, well known Artificial Intelligence techniques, such as Automated planning or scheduling, which allow us to deal with problems like resource assignment or the organisation of different activities (cost, duration, time) in a particular time period. Therefore, it will be necessary to define adequate mechanisms to translate correctly from the e-learning standards into the planning and scheduling standard language representation. In this chapter we have presented an initial approach, based on the adaptive virtual education and authoring tool used.
ACKNOWLEDGMENT

This work has been funded by the following research projects: TSI2006-12085, UAH PI2005/084, PAI-0054-4397, TSI2005-08225-C07-06 and TIN2004-03140.

REFERENCES


ADDITIONAL READING

This section provides some additional references related to the main research topics described in this chapter: AI planning and scheduling techniques, virtual education, authoring tools and e-learning standards. We have included both classical texts and some recent publications that could be used by readers to learn more about above-mentioned research themes.


AI Techniques for Monitoring Student Learning Process


Edelkamp, S., & Helmert, M. (2000). On the implementation of MIPS. In AIPS Workshop on Model-Theoretic Approaches to Planning (pp. 18-25).


Mödritscher, F., García-Barrios, V.M., & Gütl, C. (2004). The past, the present and the future of adaptive e-learning. An approach within the scope of the research project AdeLE.

Intelligent scheduling (pp. 169-212). Morgan Kaufman.


