Int. J. Engng Ed. Vol. 25, No. 6, pp. 1270–1282, 2009
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Automatic Course Redesign: Global vs. Individual Adaptation*

DAVID CAMACHO[1], ESTRELLA PULIDO[1], MARÍA D. R-MORENO[2], ROSA M. CARRO[1], ALVARO ORTIGOSA[1], JAVIER BRAVO[1]

[1] *Computer Science Department. Universidad Autónoma de Madrid, C/Francisco Tomás y Valiente, no 11, 28049 (Madrid), Spain. E-mail: {david.camacho, estrella.pulido, rosa.carro, alvaro.ortigosa, javier.bravo}@uam.es*

[2] *Departamento de Automática. Universidad de Alcalá, Ctra. Madrid-Barcelona, 28805 Alcalá de Henares (Madrid), Spain. E-mail: mdolores@aut.uah.es*

The current state of educational technologies allows the design of a new kind of tools and frameworks that can use well known Artificial Intelligence (AI) techniques, such as Planning and Scheduling, to improve some aspects of the educational process. The quality of virtual education courses can be improved by automatically detecting flaws and providing alternatives, or solutions, to the educational designers. Nowadays, most educational platforms are designed to aid educators in both designing the course contents and controlling interactions with students. We propose a new AI-based approach that could be used by educators to monitor, measure and detect problems in their deployed courses. We describe the main issues related to AI techniques and e-Learning technologies, and how Longlife Learning processes and problems can be represented, detected and managed by using an integration of planning and scheduling techniques.

Keywords: adaptive educational hypermedia; artificial intelligence; planning & scheduling; lifelong monitoring processes

INTRODUCTION

CURRENT E-LEARNING and virtual education technologies have experienced an increasing research interest thanks to the use of information technologies and the Internet [1, 2]. The use of these technologies has generated a new kind of tools and frameworks that can be used by educators to design, deploy and control courses. Several well-known e-Learning standards, such as IMS [3], SCORM [4] or LOM [5], are currently being used to define and develop new adaptive virtual-based education tools [6]. These tools support the creation of personalized learning designs (LD) [7]. These new designs make it possible to reuse and exchange useful information among different platforms. They can be used by educators (and/or course designers) not only to define the contents of a course (i.e. by using the IMS LD specification), but also to create adaptive and personalized learning flows, so that the educational system can monitor and control the whole learning process.

Most of the e-learning environments currently available contain pre-fixed courses where the user navigates and learns concepts. Some e-learning tools include Situation Learning (SL) courses where the user is presented with different predefined situations where (s)he 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 student attention. Although instructors can get statistics as well as other information about student progress, there is still a lack of feedback from previous users, about the tool, the instructors, what the user is interested in and future users. Among the tools developed on this direction we can mention the CourseVis system [8] and the Dynamic Assembly Engine [9]. An approach for automatic course generation (in some ways similar to the one presented in this paper) is the work of Ulrich [10] who uses an AI hierarchical task network (HTN) planner called JSHOP [11] that assembles learning objects retrieved from one or several repositories to create a whole course. Not only can our approach link learning objects, but it can also schedule them along a period of time and consider previous student results to generate different Learning Designs (LD). Planning and scheduling techniques have already been used in some approaches such as [12, 13]. They also use AI Planning and Scheduling techniques to automatically build a LD from the IMS-LD standard. But they do not use any deployment learning tool where students can interact or try to improve the LD using the results obtained from the students. Our approach can customize the course to each student. But the other approaches are very general and more oriented to the definition of courses and not to the students' requirements. Planning algorithms have also been used in the area of Adaptive

* Accepted 7 June 2009.

Hypermedia. In [14] reactive planning of contents for instruction makes it possible to provide adapted tutoring and coaching through plan repairs and complete replanning. In [15] planning is used for composition of Adaptive Web Services, with the goal of fulfilling dynamically changing requirements based on meta descriptions and functional properties of Web services, along with the initial 'state of the world' and the desired goal.

Here we present a new approach to the problem of control and monitoring Learning Design courses that integrates Artificial Intelligence (AI) techniques, such as Automated Planning and Scheduling, to develop an educational solving/reasoning module. These techniques are used to automatically solve problems related to course design aspects such as learning unit timing and ordering. Wrong decisions taken regarding the previous issues can be detected based on the educators/students interactions by using a statistical module. For example, the tests and exams proposed by educators can be used by a statistical subsystem to automatically detect those learning units where the success rate is under an expected threshold defined by the educators.

The methodology proposed involves the integration of an existing e-Learning platform with an AI based reasoning system [16]. In order to do so it will be necessary to map the metadata provided by the educators, as well as the results obtained from educators/students interactions, into an appropriate representation that could be used by any planner and scheduler to reason about plans. Those plans could be finally translated into a new Learning Design that the educators can check and verify. The main advantage of this approach, in comparison to other adaptive tutoring systems and e-learning platforms, is that in existing systems the adaptation is based only on current student behaviour. However, the proposed approach uses the information collected from the interactions of all the students enrolled in the course since its first deployment. In this way, adaptation evolves from an individual to a global-based adaptation.

We use a metadata-based model representation that is used to manage Learning Designs for the Higher Education courses [17]. This model is instantiated in a particular course programme, implemented by using a specific adaptive learning tool, named TANGOW (Task-based Adaptive learNer Guidance On the Web) [4]. The proposed metadata model representation allows us to specify an 'annotated' course that incorporates new semantic information related to the contents of the course. This model can be used by a planning/scheduling module, which automatically recommends several modifications in the learning module's duration and ordering. We show how, by using the selected e-learning platform and a statistical module, it is possible to automatically generate an appropriate representation that can be used by a particular integrated planning and scheduling system (named IPSS). In other words, we describe a new planning domain that can be used in e-Learning environments, and shows how it is possible to integrate planners and schedulers by using a metadata-mapping process.

TANGOW

The TANGOW system allows tutors to create Web-based courses that adapt their contents and their navigational structure to the student who accesses the course. This adaptation is based on student features (background, goals, language, . . .) as well as on interactions with the course (accessed pages, results obtained when solving exercises, time spent to understand each concept, . . .). Although adaptation involves an increased effort in the design phase of a course, from the student point of view it represents an advantage over courses whose contents and structure are static.

Basis of TANGOW

As depicted in Fig. 1, the TANGOW system has two main components:

TANGOW designer tool

This tool is used by the course designers (usually teachers) to create Web-based courses. To do this, the designer must specify the learning activities that are part of the course being designed. TANGOW uses the term teaching task to refer to a learning activity in a wide sense.

A teaching task can be a theoretical explanation about a concept, an exercise, or an example. In addition, the designer must establish an order or a sequencing among the learning activities. In TANGOW, task ordering is implemented by using rules of the form:

Ti –> T1, T2, . . . , Tn (sequencing mode, activation condition),

where Ti, T1, T2, . . . , Tn are teaching tasks.

A rule can be read as 'in order to consider task Ti as achieved by a student, tasks T1, T2, . . . , Tn

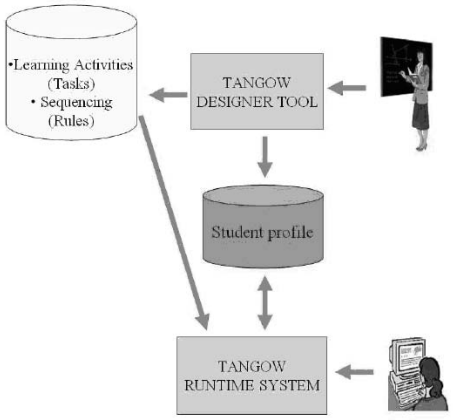

Fig. 1. TANGOW system components.

must be achieved in the order specified by the sequencing mode'. Three different sequencing modes are available: AND indicates that T1, T2, . . . , Tn must all be achieved in the order established in the rule. ANY indicates that T1, T2, . . . , Tn must all be achieved but the order is irrelevant. OR indicates that at least one of the T1, T2, . . . , Tn tasks must be achieved.

A rule can only be applied if its activation condition is fulfilled. Rule conditions can relate to the student profile (age, knowledge level, . . .), or to the actions performed by a student when interacting with a course.

The TANGOW runtime system

When a student wants to access a course, it connects to the TANGOW runtime system through the Web. The runtime system consists basically of a selection of the most appropriate tasks to be proposed to each student according to the rules that are activated when their condition is satisfied.

The courses generated with the TANGOW system are adaptive, that is, different students will be proposed different activities and will be presented with different materials each time, in a different sequence (and with different flexibility guidance) depending on their needs and preferences. Adaptation is implemented by making use of the two TANGOW course components: tasks and rules. Adaptation is achieved because rule activation conditions relate to student features as well as to their interactions with the system (i.e. concepts already mastered, exam results, solved exercises, etc.). Therefore, different activities can be proposed each time depending on those parameters, which gives rise to different course structures. In addition, for explanation of the concept corresponding to a task, different versions can be supplied by the course designer. These versions can vary according to the student language, knowledge of the course subject, or disability level (if any).

More details about the fundamentals of TANGOW and its rule-based adaptation mechanism can be found in [18].

TANGOW log files

All the information related to the interaction of each student with the system is stored in log files. These log files not only reflect all the actions the student carries on during the course, but also keep all the information needed between sessions. In that way, the TANGOW system has precise knowledge about where the student left the course during a previous visit. Log files are divided into three sections: *student profile, tasks* and *trace*.

1) Student profile. This section contains the attributes, values of the student profile. These values are constant for the course. For example, the profile section for a given student can be defined by:

```
<profile>
<feature name=''goal'' value=''detailed''/>
<feature name=''backgroundC'' value='' no''/>
<feature name=''knowledge'' value='' novice''/>
</profile>
```

The example describes a student interested in studying the subject in depth, without previous background in language C and a novice regarding data structures.

2) Tasks. This section describes the current state of already visited tasks, as well as the tasks currently scheduled by the system, according to the student profile and the course rules. For each task, the relevant attributes for this proposal are:

   – name: task id.
   – initTime: timestamp of the task starting.
   – activityType: either theoretical (T), practical (P) or example (E).
   – grade: score in the task.
   – complete: level of completeness of the task.
   – visits: number of times the student has accessed to the task.
   – parent: parent task.

For example, the following description corresponds to a practical activity (*activityType* = 'P'), already completed by the student (*complete* = '1.0') with a mark of 7/10 (*grade* = '0.7'). The student has visited this task six times (*visits* = '6').

```
<task name=''ADTTypeCExer''>
  <attribute name=''initTime'' value=''2007-
    03-28T21:34:03''/>
  <attribute name=''activityType''
    value=''P''/>
  <attribute name=''grade'' value=''0.7''/>
  <attribute name=''complete'' value=''1.0''/>
  <attribute name=''visits'' value=''6''/>
  <attribute name=''parent'' value=''ADT''/>
</task>
```

3) Trace. This last section contains the complete trace of the interaction of the student with the course. It stores information about each action taken by the student. The types of actions recorded are: task starting, task leaving or task (either complete or not) revisiting. The information recorded in each line (entry) of the trace section includes, among other data:

   – task: task id.
   – complete: it indicates the level of completeness of the task.
   – numVisits: number of visits to this task.
   – success: it indicates whether the task is considered successful or not.
   – timestamp: recorded entry time.
   – type: action of the student; the action executed can be:
     * ''START-SESSION'': beginning of the learning session; used to divide sessions.

  * ''FIRSTVISIT'': first time a task is visited.
  * ''REVISIT'': any subsequent visit to the task.
  * ''LEAVE-COMPOSITE'': the student leaves the composed task.
  * ''LEAVE-ATOMIC'': the student leaves the atomic task.
- taskType: theoretical (T), practical (P) or example (E).

As an example, the following data correspond to two log entries for the student from the previous examples:

```
<entry task=''ADTTypeCExer''
complete=''0.0'' numvisits=''2'' success=''no''
   timestamp=''2007-03-06T19:24:33''
type=''REVISIT'' activityType=''P''/>
<entry task=''ADTTypeCExer''
   complete=''1.0'' numvisits=''2''
   success=''no'' timestamp=''2007-03-
   06T19:31:43''
   type=''LEAVE-ATOMIC''
   activityType=''P''/>
```

The previous two entries reflect the situation in which a student accesses the task ''ADTTypeCExer'' for the second time (*type="REVISIT"* and *numvisits="2"*). The student has not yet completed this task successfully. In fact, at the first time of access no progress was made (*complete="0.0" and success="no"*). In addition, the timestamp of this second access is ''2007-03-06T19:24:33''.

The second entry reports the student leaving the task. The differences from the first entry are: the task was already completed but not successfully (complete="1.0" and success="no"). The timestamp (*2007-03-06T19:21:43*) reflects that the (simulated) student has used 7:10 minutes to complete the task.

EXAMPLE OF TANGOW-BASED COURSE

This course deals with Data Structures in C. The main components are tasks related to: Abstract Data Types (ADT): creation and use; Stacks; Queues; Lists; and Binary Trees.

As in any TANGOW-based course, the Data Structures course is described by means of:

1) a set of tasks,
2) a set of multimedia contents to be used for dynamic page generation (with different fragment versions, when considered appropriate),
3) a set of rules to describe the adaptation to be carried out.

As the most relevant components of this work are tasks and rules, these will be the ones described below. The specific course structure is dynamically generated for each student according to his/her features and needs.

In the sample course, three student features are considered with adaptation purposes:

1) The **goal** of the student: whether he/she prefers to receive in-depth information and to take part in different activities rather than getting a more general overview of the topics involved without participating in many activities.
2) The student's previous knowledge of data structures and abstract data types.
3) The student's background using C programming language.

The tutor can propose different tasks or provide different contents for each task depending on the student's attributes. Moreover, the organization of tasks within the course can also be different according to the users' features.

For example, the tasks to be accomplished by students wanting to learn about Abstract Data Types (ADT task) are selected according to their goal and background. Students with {*goal = general*} will be presented with activities ADTIntro (a brief introduction to the subject), ADTStrCSam (examples of data structures in C) and ADTTYpeCSam (examples of type definition in C). They will be able to access these tasks in **any** order. Figure 2a shows the tasks to be accomplished by this type of student, as well as the organization of these tasks within the course.

Regarding students who want to get more detailed information about the subject but have no previous knowledge of C language ({*goal = detailed && backgroundC = no*}), the course designer wants them to access the same introduction and, afterwards, tasks related to data structures in C (ADTStrC, which includes more subtasks related to the definition of data structures, the operations applicable to them, examples and exercises) and type definition in C (ADTTypeC, also including theory, examples and exercises about this subject), in this order (Fig. 2b).

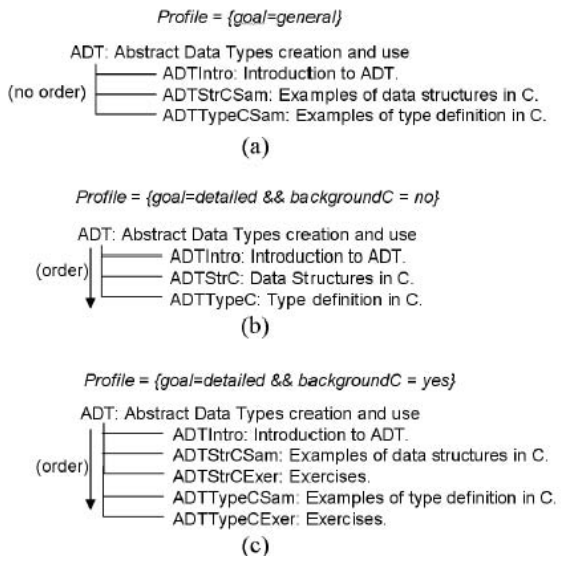

Fig. 2. Course structures desired for each type of student.

With respect to students who also want detailed information and have previous knowledge of C ({*goal = detailed && backgroundC = yes*}), the tutor wants them to skip the theoretical explanations about data structures and type definition in C. Therefore, the same introductory task will be followed by examples and exercises about data structures and type definition in C. The tutor also wants them to be directly guided through the execution of these tasks (Fig. 2c).

In order to achieve the goals described above regarding the course structure, the tutor has specified the rules shown in Fig. 3.

These rules feed the dynamic course generation mechanism that builds courses on the fly. More specifically, this set of rules led to the generation of the alternative course structures described above. For each rule, the following attributes must be specified: a composed task to be tackled; the subtasks that are part of that task; a sequencing mode from those described above; and, finally, a rule precondition, which specifies the users for whom the rule will be activated. Rule preconditions make it possible to vary the tasks to be proposed and the navigational guidance to be provided according to the user the tasks are intended for.

In another part of the course, the tasks to be presented depend on user knowledge of the subject. For example, when studying the theoretical application on stacks, the general idea of this application will be presented to novice students before the pseudo-code. However, advanced students will be directly carried to the pseudo-code. Finally, students who want general information about the whole subject will be presented with examples of different applications, while students wishing to study the subject in depth will be presented with tasks including theory, examples and exercises in each application. Tasks are defined just once, and rules represent the organization and sequencing of these tasks for different types of students. Therefore, when certain tasks appear as composed tasks or subtasks in different rules, they are in fact references to unique tasks defined just once.

The course structure generated at runtime for a novice student with previous knowledge of C, who

CONDITION	TASK	SUBTASK	SEQ.
-	DataStructures	ADT, Stack	AND
[goal= general]	ADT	ADTIntro, **ADTStrCSam, ADTTypeCSam**	ANY
[goal= detailed && backgroundC = no]	ADT	ADTIntro, **ADTStrC, ADTTypeC**	AND
[goal= detailed && backgroundC = yes]	ADT	ADTIntro, **ADTStrCSam, ADTStrCExer, ADTTypeCSam, ADTTypeCExer**	AND
[goal= general]	ADTIntro	ADTIProg, ADTIDef, ADTIObj, ADTISam	ANY
[goal= detailed]	ADTIntro	ADTIProg, ADTIDef, ADTIObj, ADTISam, **ADTExer**	AND
-	ADTStrC	ADTStrCDef, ADTStrCOper, ADTStrCSam, ADTStrCExer	ANY
-	ADTTypeC	ADTTypeCTeo, ADTTypeCSam, ADTTypeCExer	ANY

Fig. 3. Structural rules related to the ADT task and subtask.

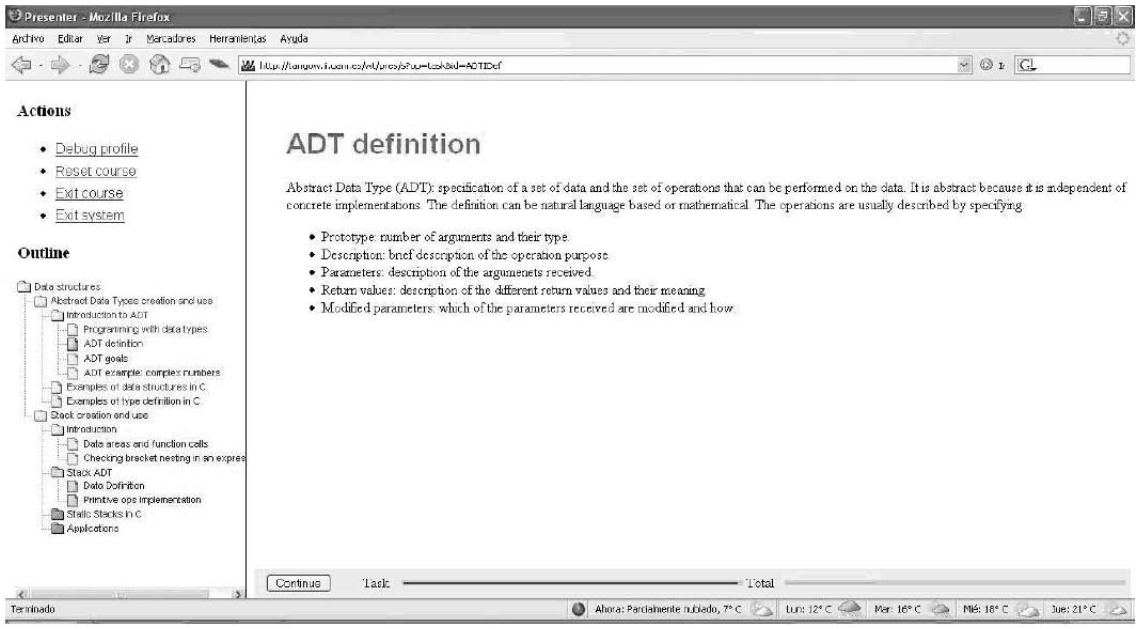

Fig. 4. Structure generated regarding ADTs with previous knowledge of C for a novice student who wants detailed information.

wants detailed information, is shown in Fig. 4. The selection of rules whose preconditions are satisfied by this student generates the annotated table of contents shown in the figure which represents the course structure. In this annotated table of contents, links to tasks are annotated according to the semaphore metaphor: a green link indicates that the task is available and its accomplishment is recommended at this time; red indicates that it is not available yet; and yellow represents that the task is available, although not recommended yet.

AUTOMATIC GLOBAL COURSE MONITORING AND REDESIGN

In order to control and automatically redesign courses into a particular learning platform, we have used IPSS [19], an AI planning/scheduling system. IPSS is able to build a solution based on the precedence relations imposed by the time and resource constraints applicable to activities (or learning tasks) implied in the problem. Each task has a set of preconditions that have to be true in order to execute the task, and a set of effects that will be true once the task has been completed. This information will be specified in the Domain file. The information related to students, teachers, specific durations of the learning tasks, etc, will be included in the Problem file. This file will be updated each time students start a course. All this information could be automatically obtained from a deployed educational tool.

The main contributions of adding a system like IPSS to an educational tool can be summarized as follows:

1) to detect flaws or inconsistencies in the learning designs such that the total time exceeds the expected course duration,
2) to propose automatically new learning designs thus saving time to teachers,
3) to consider students results in these new learning designs which will help to improve its quality. The following subsections describe how IPSS has been integrated into the TANGOW tool.

Integrating IPSS and TANGOW

The integration of IPSS and TANGOW requires a correspondence between the information, and knowledge, about students and learning designs in both systems. To achieve this, metadata extracted from the TANGOW system and the tutors who design a course, will be translated into the planning representation language, and vice versa: the solutions found by the planner will be mapped into the educational platform representation language. We have designed an integration that doesn't need interleaving (in real time) of the educational with the planning processes. Figure 5 shows the architecture of the extended educational-planning system that result from this integration. The left side of the figure represents the two initial systems (TANGOW and IPSS). The right side shows the new modules designed to map the data between both systems.

The whole monitored learning process, and therefore the integrated system, can be described in four main steps:

1) Initially, the tutor:
   – Defines the teaching tasks and rules to build the adaptive course, by using the TANGOW tool.
   – Assigns to each task a priority, and a time estimation (a high priority means that the task is considered to be essential in the learning process). Data are entered using the Priority/Time module.
   – Selects the number and kind of dimensions that will be used to generate the meta-data. In our example, the tutor has considered that the knowledge-student level is a parameter to

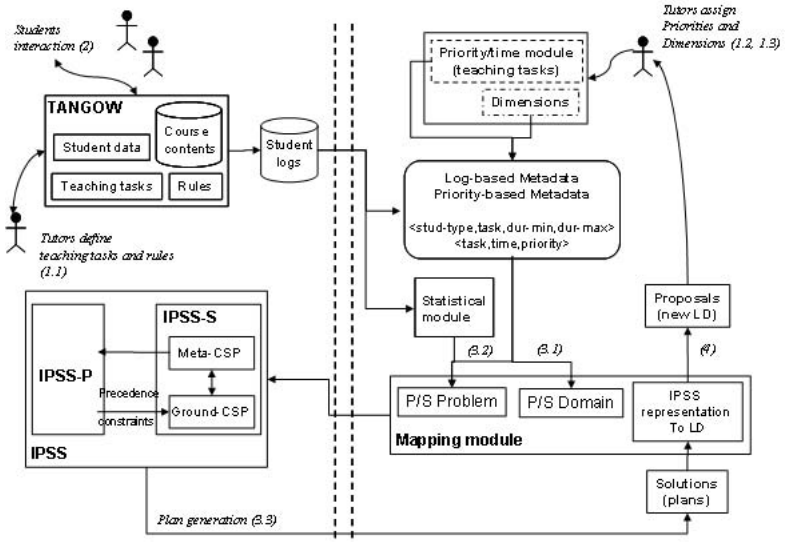

Fig. 5. Integration of TANGOW tool and IPSS system.

be taken into account for adaptation purposes.
2) Once the students have interacted with TANGOW, several log files are stored in the system. Using this information (students logs, teaching tasks, rules, task priorities, time estimation) the meta-data are generated based on logs and educator estimations.
3) Once the meta-data are obtained from the students' logs, the statistical module analyses the results (i.e. number of students that have passed a particular exam, question or course module), and the duration time for each course module. Then the mapping module proceeds as follows:
   – The meta-data, and the statistical information, are used to generate the domain and the initial state that will be used later by IPSS to solve the defined problem.
   – By using the log files and the statistical information, IPSS looks for solutions that solve possible flaws in the initial learning design by taking into account the teaching task decomposition, its priorities and estimated duration time.
   – For each student profile, a plan is generated with a possible scheduled course.
4) Finally, these solutions are mapped as new "proposals" that will be evaluated by the course designers. These proposals relate to modifying task durations, removing a particular task or reordering tasks in a particular course module.

To understand how the whole system is able to flexibly generate new solutions (or LD), once a particular problem is detected through the utilization of the statistical module, two of the previous processes need to be analysed:

1) How the IPSS-based planning domain and the IPSS-planning problem are generated from TANGOW and the statistical data.
2) How these data are handle by IPSS to generate new solutions.

IPSS-based domain to automatically redesign TANGOW courses

In the example shown above, the course designer (by using the Priority/Time module) provides the IPSS system with the metadata needed to control the course. Data are shown in Table 1.

The information showed in Table 1, provides the initial maximum time estimation for each course module (and their related submodules). We have included the estimated time for each task and subtasks in the column labelled *Simul(min)*. The simulated time required to complete each of the modules, for a group of 100 students, appears in the next column. In this example we have considered that the simulated time to finish the modules (ADTIntro, ADTStrC and ADTTypeC) has not been correctly estimated by the tutor, so the actual time required to complete them is greater than predicted. These estimations are calculated by using the average among the completion times required by all the students. The last two columns in Table 1 refer to the priority and the success rate for each module and related submodules. The priority of each task is given by the tutor whereas the success rate is obtained from the statistical module.

This information is translated into the syntax required by IPSS to decide which task could be modified. These modifications are usually related to time duration (i.e. to reduce or increase the time duration of a particular task) or ordering among tasks. All the IPSS operators in the domain are created dynamically from the tasks defined in TANGOW. Consequently, each course will have a different domain definition. To show how the IPSS operators are generated let us consider the ADTTypeC task of Table 2 extracted from Fig. 3.

This task is composed of three subtasks which are represented in the set of preconditions of the IPSS operator to be applied. Other preconditions depend on:

1) the priority of the instantiated task being lower or equal than the base priority defined by the educators,

Table 1. Time duration and priority for ADT course

No.	Task	Max. Time (min.)	Simul. (min.)	Priority	Success (%)
1	ADTIntro	Adding tasks 2 to 6 = 135	150	1	
2	ADTIProg	10	30	3	
3	ADTIDef	35	30	2	
4	ADTIObj	10	30	5	
5	ADTISam	40	30	1	
6	**ADTIExer**	**40**	**30**	**4**	**80%**
7	ADTStrC	Adding tasks 8 to 11 = 140	130	2	
8	ADTStrCDef	20	30	4	
9	ADTStrCOper	30	40	2	
10	ADTStrCSam	40	30	1	
11	**ADTStrCExer**	**50**	**30**	**3**	**50%**
12	ADTTypeC	Adding tasks 13 to 15 = 80	90	3	
13	ADTTypeCTeo	20	30	2	
14	ADTTypeCSam	20	30	1	
15	**ADTTypeCExer**	**40**	**30**	**3**	**60%**

Table 2. ADTTypeC Task and Subtasks

Condition	TASK	SUBTASK	SEQ.
—	ADTTypeC	ADTTypeCTeo, ADTTypeCSam, ADTTypeCExer	ANY

2) the duration of the task based on the tasks already inserted in the plan and
3) the number of students that have achieved the task is equal or higher than the base percentage defined by the educators.

The operator has only one effect, that the student learns the task.

Regarding the problem definition, the first part of Fig. 6 shows some initial conditions for the problem. This information is automatically taken from the data introduced by the educators, such as priorities and relationships among tasks and subtasks, from the log files that contain task durations, and form the tests solved by the students that indicate the number of students that have passed or failed a specific task.

The second part of Fig. 6 shows the high level goals that IPSS needs to accomplish. In this case, there are two goals: that Mary and Anthony learn the ADT course. These two meta-goals will be decomposed into lower level goals, that is, for Mary and Anthony to learn the ADT course they need to learn each course unit. Then, these two meta-goals are translated into the following goals:

(learnt Mary ADTIntro) (learnt Mary ADTStrc)(learnt Mary ADTTypeC) . . .
(learnt Anthony ADTIntro) . . . (learnt Anthony ADTTypeC) . . .

Generating new proposals

The previous example, and the corresponding simulated times (see Table 1), have been implemented to show how IPSS is able to generate solutions when there is not enough time to execute all the teaching tasks. In this situation the process is as follows:

1) It looks for OR rules that have not been executed, and removes the subtasks with a lower priority (see Fig. 7).
2) If no OR-rules are available, an ANY-rule is selected and its (N) subtasks are ordered by using its priorities (ANY-rules are ordered by their relative time precedence, those rules that will be immediately executed are selected first):
   a. Higher priority subtasks are scheduled and executed.
   b. If there is not enough time to execute all the subtasks, new ANY-OR-rules are generated. This new kind of rules is generated by incorporating the N-1 subtasks with a higher priority in the ANY-rule into a new ANY-rule, and by creating a new OR-rule with taskN (the task with a lower priority in the original ANY-rule (see Fig. 7).
3) Once the ANY-OR-rule is generated, it is applied and a new course schedule is created. This new schedule is created by removing the lower priority subtask from the ANY-rule (see Fig. 7).
   a. If the new proposed course schedule is adequate, this solution is given to the teachers for their analysis and approval.
   b. Otherwise, a new ANY-OR-rule is generated as follows: the ANY-OR-rule is transformed into a new ANY-rule (now with N-2 subtasks) and a new OR-rule (that includes taskN-1). This process is recursively performed until a solution is found (therefore, the ANY-OR-rules generation stops when a teaching plan that can be scheduled in the available time for the course is found).

From the example in Fig. 7, IPSS will suggest that the course designer add extra time to a task or remove some of the subtasks because the students need too much time to complete them and some (high priority) tasks could be missing. Therefore, if no extra time can be added to the introductory task, some fewer priority subtasks could be removed or summarized.

```
(initial state
   (and (base-priority 2)
        (compose ADTTypeC ADTTypeCTeo ADTTypeCSam ADTTypeCExer)
        (duration ADTTypeC (80 90))
        (priority ADTTypeC 3)
        (priority ADTTypeCTeo 2)
        (priority ADTTypeCSam 1)
        (priority ADTTypeCExer 3)
        (sucess ADTIntro 80)
        (sucess ADTStrC 50)
        (sucess ADTTypeC 60)
        (base-success 45)
         ...
    ))
(goal (and (learnt Mary ADT)
           (learnt Anthony ADT)))))
```

Fig. 6. Problem definition.

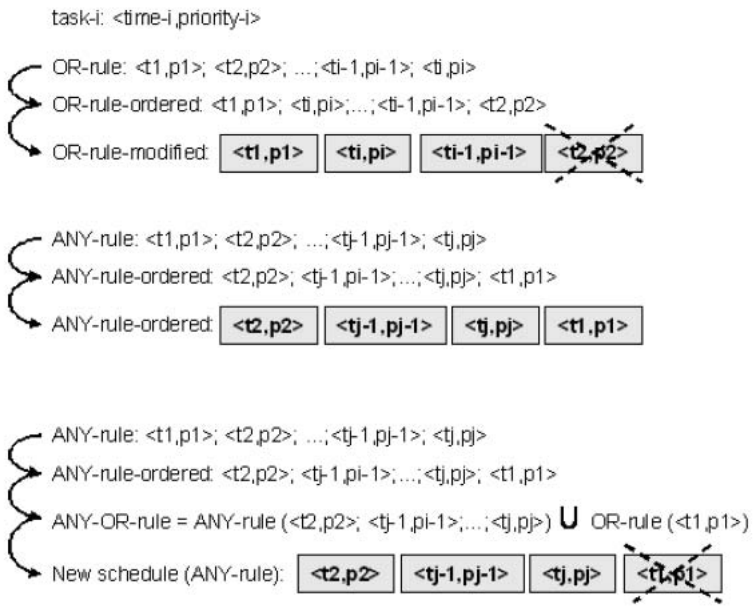

Fig. 7. TANGOW rules processed by the IPSS system, and new rule generation (ANY-OR).

In the previous example (see Table 1) some tasks have been overestimated (ADTStrc task), or underestimated (ADTIntro and ADTTypeC tasks); then IPSS (and the new proposed rules) can suggest that the course designer reduce the total time assigned to lower priority subtasks (i.e. ADTStrCDef and ADTStrCExer) to reassign this extra time to other underestimated tasks. However, this extra time is not enough to complete the whole course (this is a problem because we could miss some important subtasks in the course). Although 10 minutes have been gained, we need 15 minutes to complete the first task (150 –135) and 10 minutes to complete the last one (ADTTypeC, 90–80). Consequently, it is necessary to redesign the course to obtain about 15 minutes more. Now, the estimated success of different tasks is used to recommend the reduction of a particular module in the course. In our example, the first task (ADTIntro) has a very high rate of success (80%). For this reason the lower priority submodules (ADTIExer and ADTIObj) could be reduced (in time duration). Therefore, IPSS will suggest to the course designer a reduction of both subtasks. The planner distributes these reductions by applying the following criterion (heuristic): 'the higher the priority the lower the reduction in time'. If we assume that the course designer has assigned five minutes as the minimum time for the ADTIObj task, the new proposed design of this module is shown in Table 3.

Table 3. New proposed design for ADTIObj task

No.	Task	New estimated time
1	ADTIntro	Adding tasks 2 to 6 = 120
2	ADTIProg	10
3	ADTIDef	35
4	ADTIObj	5 (min. time allowed)
5	DTISam	40
6	ADTIExer	30

EXPERIMENTAL EVALUATION

To test the performance of the proposed approach, data were first needed from student interaction with the TANGOW system. Afterwards the statistical module would analyse these data and propose a new course structure to course designers. With this intention, it was decided to use synthetic logs, generated from a real course, to feed data into IPSS; finally, suggestions generated by IPSS were evaluated by experts (teachers) and provided feedback about its behaviour.

Synthetic logs

Tools for improving course design based on log analysis, such as the one presented in this paper, generally do not interact directly with the delivery system or adaptation engine (TANGOW in this case), but through the generated logs. For this reason, they can be built independently from the delivery system and also from the method used to generate the log files. Synthetic logs (that is, logs generated by a program rather than resulting from the real interaction of users with the system) provide developers and evaluators of log-based tools with a practical way of testing these tools, saving time and resources. This is especially useful when it is not possible or convenient to test the tools with final users, i.e. testing an adaptive course before it is delivered to students. In these cases it is useful to utilize synthetic logs for testing how a tool works. SIMULOG (SIMulation of User LOGs) [20] can be used for this purpose

Operator-Name	Start-Time	End-Time
(OP_ADTIntro Mary ADTIntro)	0	120
(OP_ADTIProg Mary ADTIProg)	0	10
(OP_ADTIDef Mary ADTIDef)	10	45
(OP_ADTIObj Mary ADTIObj)	45	50
(OP_ADTISam Mary ADTISam)	50	90
(OP_ADTIExer Mary ADTIExer)	90	120
...		
(OP_ADTStrC Mary ADTStrC)	120	250
...		
(OP_ADTTypeC Mary ADTTypeC)	250	330
...		
(OP_ADTIntro Anthony ADTIntro)	0	120
(OP_ADTIProg Anthony ADTIProg)	0	10
...		
(OP_ADTIExer Anthony ADTIExer)	90	120
...		
(OP_ADTStrC Anthony ADTStrC)	120	250
...		
(OP_ADTTypeC Anthony ADTTypeC)	250	330
...		
(OP_App3 Anthony App3)	455	475
...		
(OP_App4 Anthony App4)	455	465

Fig. 8. IPSS result.

because it generates log files similar to the files recorded when students interact with the TANGOW system. It reads the course description and, based on a randomly generated student profile, reproduces the steps that a student with this profile would take when following a course. Student profiles are generated by using a random function that follows a Gaussian distribution, based on user defined parameters. For example, let us assume that the SIMULOG user defines that 70% of the generated logs would correspond to students with *language=''English*" and the remaining 30% with *language= ''Spanish*". If 200 students have to be simulated, SIMULOG will generate 200 log files, 140 of which will correspond to students with *language="English*" on average.

It is worth mentioning that SIMULOG mimics the decisions taken by the adaptive system. For example, if the course description contains a rule stating that for 'young' students, activity A1 is composed of subactivities S1 and S2, which will have to be tackled in this order (S1 before S2), after recording a visit to activity A1, SIMULOG will record visits to activities S1 and S2, respectively. The user can specify the distribution for every attribute (dimension) defined at the course description, and distributions will be combined to generate each student profile. For example, if user profiles with 90% of the instances having *language="English*", 90% having *experience="novice*" and 90% having *age="young*" were defined, the expected outcome would be that around 73% of the generated logs would correspond to a user profile with *language= ''English*", *experience="novice*" and *age="young*", on the average.

Experimental setup and results

In order to test the advantages of the proposed approach, synthetic logs simulating the interaction of 100 students with the system have been generated. All of them have a profile corresponding to *goal=detailed*, *backgroundC=no*. Regarding the *knowledge* attribute, 50% have *knowledge=novice* while the other 50% have *knowledge=advanced*. Simulated times for completing tasks are generated accordingly to a normal distribution with a mean = 30 minutes. In this way, on the average, 75% of the tasks are completed in 18.5–41.5 minutes. With respect to the course designer estimation, as is shown in Table 1 above, these numbers indicate that most of the students spend more time than anticipated on tasks 1 to 6, while most of them stay within the estimated limits for tasks 7 to 15.

Exercise tasks have an attribute indicating whether they have been solved successfully or not. In the example course there are three practical tasks for the simulated profile. For the ADTExer and ADTTypeCExer tasks the logs show that only 20% of the students failed them. As for the ADTStrCExer task, the logs show that 50% of the students failed. This fact indicates a potential problem in this task. IPSS generates a plan with the sequence of operators that transforms the initial state into a goal state where the start and end times are satisfied. Figure 8 shows the solution generated from the initial conditions in Fig. 6. In the solution, IPSS instantiates each operator by giving value to its different variables: the student, the tasks, the subtask list that composes the task, the duration, etc.

The fact that some (sub)tasks (operators instantiated) have prerequisite relationships with others already performed, imposes the restriction that the starting time of a (sub)task must be later in time than the ending time of its prerequisite (sub)tasks. This is the case of the ''OP ADTIDef" and ''OP ADTIObj" operators in Fig. 8. The starting time of ''OP ADTIObj" is equal to the ending time of ''OP ADTIDef". Since the Sequencing mode is ''AND", all the tasks must be performed sequentially. But these prerequisite relationships may not exist between other tasks such as the ''OP App3" and ''OP App4" tasks that can be executed in parallel.

By using the previously described experimental setup and results, six teachers with teaching experience in courses based on "Data Structures", "Programming in C" or "Structured and Object Oriented Programming" have evaluated the usefulness of this approach. All of these educators were requested as 'possible' coordinators of this course, to analyse the solutions produced by the scheduling algorithm and to compare them with the initial course planning. Table 4 shows their evaluation,- which was carried out by using a questionnaire of five basic questions, with a range of values from 1 to 7 (from worst to best), related to the quality and usefulness of the solutions given by the system.

They found the new generated courses (modifications and suggestions from the IPSS system) appropriate with a satisfaction level of 6,74 (up to 7) and with a standard deviation (s.d.) of 8,7%, and especially useful the automatic capacity of the system to predict the time to be spent on each task and to be able to modify the tasks proposed to each student at each time according to this dynamic information (6,23 with a s.d. of 18,4%).

Table 4. Educators (course coordinators)' evaluation of IPSS solutions

No.	Question	Mean value [1..7]	Standard Dev. (%)
1	Possibility to extend this approach to other kind of courses (different from computer science, or programming courses)	6,83	3,4
2	Capacity to detect problems in the educational process	6,23	18,4
3	Usability of the framework. Easiness to understand and manage the solutions provided by the planner. Easiness to integrate the suggestions into a real course program	3,22	32,6
4	Quality (number and precision) of modifications suggested by IPSS. How much of them are really closer to a real educator decision?	5,87	11,3
5	Usefulness of modifications and suggestions generated from the system. How much of them could be really incorporated in the course?	6,74	8,7

The whole quality of the proposed courses, taking into account how many of those modifications could be used by the educators in a real course, was 5,87 (s.d. 11,3%), which indicates that the main suggestions would be incorporated in the final program course.

They highlighted the usefulness of the integration of IPSS and TANGOW in the sense that this makes it possible to help students to learn on their own not only by considering their personal features, needs and task accomplished, but also time predictions and real time measures, helping them to better organize their learning tasks. Therefore, most of the educators agreed about the possibility of using this approach in other kind of courses (6,83 with a s.d. 3,4%), because the current approach is both, independent of the course contents (it uses statistical information to detect problems), and domain independent to propose solutions. The worst results in our evaluation were related to the usability capabilities of our current approach (3,22 with a s.d. 32,6%), although some educators found the output of the planner (most of them had previous experience in AI techniques) usable, the current situation is that an automatic mapping from these outputs are necessary to alleviate the work of understanding and reusing the information obtained from the planner.

CONCLUSIONS

We have presented a new approach to control and monitor Learning Design through the integration of planning and scheduling techniques in a particular Web-based adaptive learning tool (TANGOW). The system can be described as a new e-learning platform that integrates a solving-reasoning module. This module is able to solve problems related to course design aspects such as task timing and ordering.

The proposed methodology requires several mapping processes to correctly translate the data (and metadata) provided by the tutors, and the results obtained from tutors/students interactions, into an appropriate representation that can be used by the reasoner system (IPSS). The generated plans will finally be translated into a new Learning Design, in the form of "proposals" that can be checked and verified by the course designer.

Decisions taken by the solving-reasoning module regarding task timing and ordering can also be evaluated, and wrong decisions can be detected by data mining techniques that analyse student interactions within an adaptive course. For example, student results in tests and exams can be analysed to automatically detect learning units where the success rate is under an expected threshold or inappropriate task sequencing, among others. In this sense, it is possible to detect situations in which plans generated by IPSS+TANGOW lead to bad results or provoke undesirable user behaviours (i.e. students jumping from one task to another unexpectedly, students getting noticeably bad results in practical tasks, students abandoning the course, etc.). To detect undesirable situations, an evaluation module based on data mining techniques will be incorporated into the proposal. Some work has already been done in this direction, such as [21], with the aim of automatically evaluating adaptive courses in TANGOW, or [22], in which automatic recommendations are given based on Web usage mining techniques.

The main advantage of our approach, in comparison to other adaptive tutoring systems and e-learning platforms, is that in most existing systems the adaptation is based only on the current student behaviour. However, the proposed approach uses the information collected from the interactions of all the students enrolled in the course since it was deployed. In this way, the adaptation evolves from being an individual-based to a global-based one.

Acknowledgments—This work has been funded by the research projects TIN2007-64718, TIN2008-02729-E/TIN, TSI 2006-12085, and PEII09-0266-6640. The authors wish to thank the anonymous referees for their careful reading of the manuscript and their fruitful comments and suggestions.

REFERENCES

1. R. Kozma, Learning with media. *Rev. Educ. Research*, **61**(2), (1991), pp. 179–212.
2. C. Schmitz, S. Staab, R. Studer, G. Stumme and J. Tane, *Accessing Distributed Learning Repositories through a Courseware Watchdog*. Proceedings of the E-Learn 2002—World Conference on ELearning in Corporate, Government, Healthcare for Higher Education, (2002), pp. 909–915.
3. IMS-LD. Ims learning design. IMS Global Learning Consortium. http://www.imsglobal.org/learningdesign/index.html (2006).
4. SCORM. Sharable courseware object referencemodel. http://www.academiccolab.org/projects/scorm.html (2006).
5. LOM. Learning object metadata. http://ltsc.ieee.org/wg12/ (2006).
6. M. Specht and D. Burgos, Implementing Adaptive Educational Methods with IMS Learning Design. *Lecture Notes in Learning and Teaching,* (2006), pp. 241–251.
7. R. Koper, Learning Design. A Handbook on Modeling and Delivering Networked Education and Training, Springer-Verlag, (2005), pp. 3–20.
8. R. Mazza and V. Dimitrova, CourseVis: *Externalising student information to facilitate instructors in distance learning*. Proceedings of the International conference in Artificial Intelligence in Education, U. Hoppe, F. Verdejo, and J. Kay, (Eds). (2003), pp. 117–129.
9. R. Farrell, S.D. Liburd and J.C. Thomas, *Dynamic Assembly of Learning Objects.* Proceedings of 13th International World Wide Web Conference, (2004), pp. 117–129.
10. C. Ullrich, *Course generation based on htn planning*. Proceedings of 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems, (2005), pp. 74–79.
11. O. Ilghami and D.S. Nau, *A general approach to synthesize problem-specific planners*. Tech. Rep. Technical Report CS-TR-4597, Department of Computer Science, University of Maryland (2003).
12. A. González-Ferrer, L. Castillo, J. Fernández-Olivares and L. L. Morales, *Workflow Planning for E-learning Center Management*. Proceedings of 8th IEEE International Conference on Advanced Learning Technologies (ICALT 2008), IEEE Computer Society Press (2008).
13. L. L. Morales, L. Castillo, J. Fernández-Olivares and A. González-Ferrer, Automatic Generation of User Adapted Learning Designs: An AI-Planning Proposal. Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems (AH2008), *Lecture Notes on Computer Science* **5149**, (2008), pp. 324–328.
14. J. Vassileva, *Reactive Instructional Planning to Support Interacting Teaching Strategies*. Proceedings of the 7th World conference On AI and Education, (1995) pp 334–342.
15. I. O'Keeffe, O. Conlan and V. Wade, A Unified Approach to Adaptive Hypermedia Personalisation and Adaptive Service Composition. Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2006), *Lecture Notes in Computer Science* **4018,** (2006) pp. 303–307.
16. D. Camacho, A. Ortigosa, E. Pulido and M.D. R-Moreno, *Advances in E-Learning: Experiences and Methodologies.* Information Science Reference, formerly Idea Group Publishing, (2008) pp. 149–172.
17. D. Camacho and M.D. R-Moreno, Towards and automatic monitoring for higher education learning design. *Int. J. Metadata, Semantics and Ontologies* (IJMSO) **2**(1), (2007), pp. 1–10.
18. R. Carro, E. Pulido and P. Rodríguez, Dynamic generation of adaptive internet-based courses. *J. Network and Computer Applications*, **22**, (1999), 249–257.
19. M. D. R-Moreno, A. Oddi, D. Borrajo and A. Cesta, IPSS: a hybrid approach to planning and scheduling integration. *IEEE Transactions on Knowledge and Data Engineering* **18**(12), (2006), pp. 1681–1695.
20. J. Bravo and A. Ortigosa, *A validating the evaluation of adaptive systems by user profile simulation. Proceedings of Fifth Workshop on User-Centred Design and Evaluation of Adaptive Systems* held at the Fourth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2006), Ireland (2006), pp. 479–483.
21. C. Vialardi, J. Bravo and A. Ortigosa, Improving AEH Courses through Log Analysis. *J. Universal Computer Science* **14**(17), (2009), pp. 2777–2798.
22. M. K. Khribi, M. Jemni and O. Nasaraoui, *Automatic Recommendations for E-Learning Personalization Based on Web Usage Mining Techniques and Information Retrieval.* Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies (ICALT 2008), (2008), pp. 241–245.

David Camacho is currently working as Associate Professor in the Computer Science Department at Universidad Autónoma de Madrid (Spain). He holds a Ph.D. in Computer Science (2001) from Universidad Carlos III de Madrid, and a B.S. in Physics (1994) from Universidad Complutense de Madrid. He has published over 50 journal, books, and conference papers. His research interests include Multi-Agent Systems, Distributed Artificial Intelligence, Semantic Web Technologies, Automated Planning and Machine Learning.

Estrella Pulido received a degree in Computer Science in 1989 from the Universidad Politécnica de Madrid and was working in industry from 1987 until 1991. She obtained an MSc in Artificial Intelligence with honours from the University of Bristol in 1992 and a Ph.D. from the same University in 1996. She has worked at the Escuela Politécnica Superior of the Universidad Autónoma de Madrid since October 1996 where she has held an associate professor position since July 2000. She has participated in several national and international research projects and has authored many scientific and technical publications. Her main research areas include the use of virtual worlds in education, semantic Web and Web databases.

Maria Dolores R-Moreno received an M.S. degree in Physics from the Universidad Complutense de Madrid and a European Ph.D. in Computer Science from Universidad de Alcalá. She subsequently spent one year at NASA Ames Research Center as a postdoc, and nine weeks at ESA's European Space Research and Technology Centre (ESTEC). Her research interest focuses on the application of AI techniques to real domains such as satellites, workflow or e-learning. She is currently Associate Professor in the Computer Science Department at the Universidad de Alcalá.

Rosa M. Carro got her Ph.D. in 2001 at the Universidad Autónoma de Madrid and is associate professor in this university. Her research focuses on adaptive hypermedia, user modelling, mobile learning and collaborative systems. She did research on adaptive educational games in the University of Aveiro in 2001/02. She joined the Group of Applied Informatics—Cooperative Systems of the Technical University of Munich in 2002. She is a member of several associations and committees, and has co-organised several international workshops. She is reviewer for international journals, and co-author of more than 70 publications related to her research areas.

Alvaro Ortigosa got his Master in Science degree from the UFRGS-Brazil in 1995, and his Ph.D. in 2000 from the Universidad Autónoma de Madrid, where he is currently associate professor. His research focuses on adaptive system evaluation and user modelling and profiling. He joined the Group of Applied Informatics—Cooperative Systems of the Technical University of Munich in 2002. He is reviewer for international conferences related to his research areas. He has participated in more than 10 international and national projects and has co-authored more than 50 publications.

Javier Bravo is currently finishing a Ph.D. in Computer Science at Universidad Autónoma de Madrid (Spain). He holds a B.S. in Statistics (2000) at Universidad Complutense de Madrid (Spain), B.S. in Computer Science (2004) and a M.S. degree in Computer Science (2006) at Universidad Autónoma de Madrid. He also collaborated with relevant researcher groups of some institutions in the e-learning area, e.g. TELECOM Bretagne (2007) and University of Pittsburgh (2008). His research interests are focused on Data Mining and Educational Hypermedia Systems.