Interoperable Intelligent Tutoring Systems as Open Educational Resources

Gustavo Soares Santos and Joaquim Jorge, *Senior Member*, *IEEE Computer Society*

Abstract—Because of interoperability issues, intelligent tutoring systems are difficult to deploy in current educational platforms without additional work. This limitation is significant because tutoring systems require considerable time and resources for their implementation. In addition, because these tutors have a high educational value, it is desirable that they could be shared, used by many stakeholders, and easily loaded onto different platforms. This paper describes a new approach to implementing open-source and interoperable intelligent tutors through standardization. In contrast to other methods, our technique does not require using nonstandardized peripheral systems or databases, which would restrict the interoperability of learning objects. Thus, our approach has the advantage of yielding tutors that are fully conformant to e-learning standards and that are free of external resource dependencies. According to our method, "atomic" tutoring systems are grouped to create "molecular" tree structures that cover course modules. In addition, given the interoperability of our technique, tutors can also be combined to create courses that have distinct granularities, topics, and target students. The key to our method is the focus on assuring what defines a tutor in terms of behavior and functionalities (inner loops and outer loops). Our proof of concept was developed using SCORM standards. This paper presents the implementation details of our technique, including the theoretical concepts, technical specifications, and practical examples.

Index Terms—Computers and education, computer-assisted instruction, computer-managed instruction, distance learning

1 Introduction

Learning technologies are currently applied in many institutions around the world. Computer-based educational platforms, such as personal learning environments and learning management systems (LMSs), are now relatively common in our schools and universities [1], [2]. These platforms play an important role in regard to the distribution of didactic material. However, they have been used mainly to deliver noninteractive and nonintelligent educational content, such as Power Point presentations, Word documents, and Portable Document Format files [3].

Intelligent tutoring systems (ITSs) are interactive educational systems that are built by combining from artificial intelligence techniques, and concepts from the learning sciences. These systems proved to be beneficial for learning in several domains, from programming languages [4] and middle school math [5], to physics [6] and military applications [7]. Unfortunately, because of interoperability issues, ITSs cannot be loaded into most educational platforms that are currently available and that require dedicated nonstandard frameworks.

Since tutors are not interoperable, they end up not being shared and, thus, are not accessible to many users. This limitation is significant because tutoring systems can be very effective. Additionally, given that ITSs require a substantial amount of time and resources to be implemented, it is desirable that they could be shared, used by many students, accessed from many places, and loaded onto different platforms.

To increase the accessibility of ITSs, we have developed an approach for implementing interoperable tutors with the support of standards [8], [9], [10]. Specifically, we target the sharable content object reference model (SCORM) e-learning standards. Our method allows implementing web-based ITSs as learning objects (LOs) and using a novel structural design that focuses on supporting the essential features of intelligent tutors, the inner loop and the outer loop.

The main objective of this paper is to present our new approach to implement tutors as interoperable open educational resources (OERs). One of the contributions is an innovative architecture that introduces two new concepts for developing ITSs: 1) atomic tutors (ATs) and 2) molecular tutors (MTs) [11]. We define ATs as small tutoring systems that represent single activities. ATs are grouped to create MTs, which are larger LOs that represent complete educational units capable of performing task selection. Based on e-learning standards, our approach provides the means to support open source, adaptive and intelligent LOs that can be obtained from repositories of educational content, grouped if and when necessary, and loaded onto different platforms.

2 Background and Related Work

To facilitate the explanation of our approach for interoperable tutors, this section covers important topics that concern the state of the art. Section 2.1 recapitulates a few concepts that are associated with the SCORM e-learning standards.

- *G.S. Santos is with the Department of Information Systems and Computer Engineering, Instituto Superior Técnico, Technical University of Lisbon, Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal. E-mail: gustavossantos@ist.utl.pt.*
- *J. Jorge is with the INESC-ID Associate Laboratory, Rua Alves Redol 9, 1000-029 Lisbon, Portugal, and the Department of Information Systems and Computer Engineering, Instituto Superior Técnico, Technical University of Lisbon, Av. Rovisco Pais 1, 1049-001 Lisbon, Portugal. E-mail: jaj@inesc.pt, jorgej@ist.utl.pt.*

Manuscript received 29 Aug. 2012; revised 18 Jan. 2013; accepted 20 Apr. 2013; published online 26 Apr. 2013.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2012-08-0118.
Digital Object Identifier no. 10.1109/TLT.2013.17.

Section 2.2 explains ITS concepts that are significant for our implementation method. Finally, Section 2.3 discusses related work.

2.1 SCORM Standards

The SCORM is a set of standards and specifications for web-based learning [12]. SCORM is developed and maintained by the advanced distributed learning (ADL) initiative [13]; however, SCORM is a product of several entities, such as IEEE, AICC, Ariadne, and IMS Global. SCORM specifies three main components: 1) the runtime environment specification (RTE); 2) the content packaging specification; 3) the sequencing and navigation specification. These specifications provide guidelines that define, in practice, how educational content should communicate at runtime, how educational content should be packaged and described and how educational content should be sequenced and navigated. SCORM is the de facto standard for learning technologies and according to ADL, its key benefits include interoperability, accessibility, reusability, and durability [14].

2.2 Intelligent Tutoring Systems

Intelligent tutoring systems are educational systems that can engage students in interactive reasoning activities that require a deep understanding of the domain being taught and that also require considerable comprehension of students' behaviors. Intelligent tutors usually employ theories of learning by doing [15] and can also apply a series of different technologies for implementation.

The classic architecture of a tutoring system comprises four elements or modules [16], [17], [18]. The traditional instructional model of an ITS is based on students engaging in problem solving activities through a user interface. The domain module (typically an expert system) evaluates the actions that are performed by the students. The student model records what the ITS knows about the students and the pedagogical module provides instructional interventions and feedback to the apprentices.

This traditional view of ITSs is still very accepted by the community. However, recent papers stress functionality over structure [19], [20], [21], describing ITSs as having two main loops [21]: 1) the inner loop and 2) the outer loop. The inner loop is responsible for providing personalized feedback, hints, and direct problem solving assistance to students. The inner loop also assesses students' competence and registers it on the student model. Using the information that is obtained about the student, the outer loop performs task selection. Pseudo Code 1 illustrates this functional view of ITSs.

This functional approach to ITSs does not directly challenge the traditional modular decomposition because it is devoid of recommendations regarding the internal structures that model the intelligent behavior of a tutor. However, this view contributes significantly to the literature by providing important guidelines on how ITSs should behave [21]. It defines the key functionalities that must be implemented by both loops. Furthermore, it describes services to assist students with step-based problem solving, including minimal feedback, next step hints, error-specific feedback, assessment of knowledge and review of complete solutions.

Pseudo Code 1. ITS Loops.

```
until tutoring is complete, repeat {
      tutor selects a task;
      until task is complete, repeat {
         student executes a step;
         tutor may present a hint;
         tutor updates the student model;
         tutor presents feedback on the step;
      }
      student submits the solution for the task;
   }
```

2.3 Related Work on the Interoperability of ITSs

Previous research on interoperable and adaptive educational systems has already presented some excellent results [22], [23], [24], [25]. Project GRAPPLE focused on integrating LMSs with adaptive learning environments, by developing an architecture for a generic adaptive webserver, a browser-based authoring environment, and a distributed user modeling framework. GRAPPLE can be used for creating and serving web-based adaptive educational software. GRAPPLE supports some types of adaptation, such as content, link, and presentation; in addition, it has the capacity to support several types of user model information.

Project T-MAESTRO is an ITS that constructs learning experiences by using semantic knowledge about the students. This project also provides an authoring tool, which allows instructional designers to create adaptive courses with minimal technical expertise. T-MAESTRO is implemented using SCORM, but it defines an extension of the ADL SCORM standards using external services to support adaptation.

While these projects provide distinct methods of integrating adaptive learning into LMSs, both approaches have a similar drawback concerning interoperability, which is the usage of nonstandard external systems and databases to support adaptation.

Regarding SCORM, ADL does not forbid the use of external resources for the development of standardized LOs [12]. On the other hand, ADL makes it clear that tying standardized LOs to nonstandard peripheral systems compromises the interoperability of the application. If an LO is dependent on an external system, it cannot always be shared, used, or widely distributed because of the impossibility of stakeholders having full control of the peripheral systems. Additionally, routine tasks, such as system maintenance, adding user accounts, managing concurrent accesses, scalability and many others, can be out of the control of the stakeholders. If an external system is discontinued, updated, or simply crashes, then dependent applications might stop working. Furthermore, interoperable LOs are expected to be open educational resources, and therefore, their source code should be available. However, using external systems can preclude access to parts of the source code. Therefore, updates and improvements in the educational software are limited to the available parts of the code.

In contrast to other approaches, our implementation method does not require the extension of standards by using peripheral nonstandard systems or databases. As a

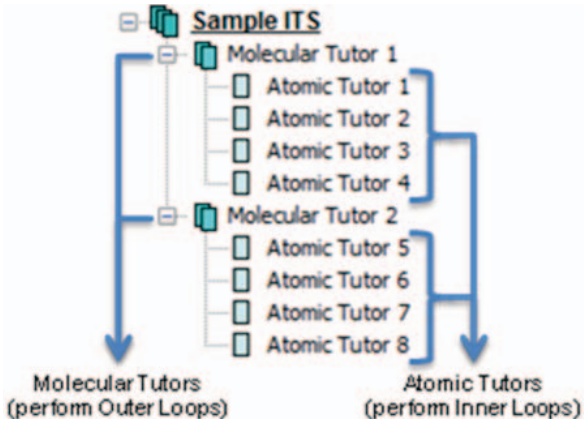

Fig. 1. Tree structure of a sample ITS.

result, our technique has the significant advantage of producing tutors that are completely interoperable, open source, and free of external dependencies.

Another important feature of our approach is that it does not require mapping ITS modules to SCORM modules. Mapping modules was conceptually suggested before [3]; however, earlier suggestions did not provide concrete evidence that direct mapping is possible or even appropriate. Indeed, our approach does not require modules. Instead, we adopt user, task, and pedagogical models [8], [9], [10], which we describe in the next section.

3 Our Approach to Interoperable ITSs

Our approach to developing interoperable ITSs as OERs through e-learning standards builds on what defines a tutor in terms of behavior and functionality: inner loops and outer loops. Accordingly, our method focuses on guaranteeing the correct behavior of these loops. Technically, what makes the approach possible is both a special software architecture and the SCORM specification. To implement the loops, we use SCORM constructs, such as sequencing definitions and tracking data. First, the tracking data, which are a standardized database for learning purposes, store our student model, which is based on proficiency levels of skills. Second, the sequencing definition uses the tracking data to perform task selection.

Our new architecture for supporting interoperable ITSs organizes the tutors into tree structures that assemble two different constructs, as shown in Fig. 1: 1) atomic tutoring systems (leaves) and 2) molecular tutoring systems (nodes). ATs must be grouped appropriately to create MTs [11]. As a consequence, because of the interoperability of the MTs, they can be combined to create other tutors and can be used in educational platforms to categorize courses on distinct educational topics and to target students. We should stress that the concepts of AT and MT are currently being introduced for the first time in the literature, and they relate exclusively to our approach.

First, ATs are similar to small tutoring systems that are responsible only for performing inner loops. Therefore, they represent instructional tasks, exercises, and small assignments. This scope is essential because ITSs are directly associated with step-based problem solving and ATs implement this functionality. Second, MTs are aggregate ATs and

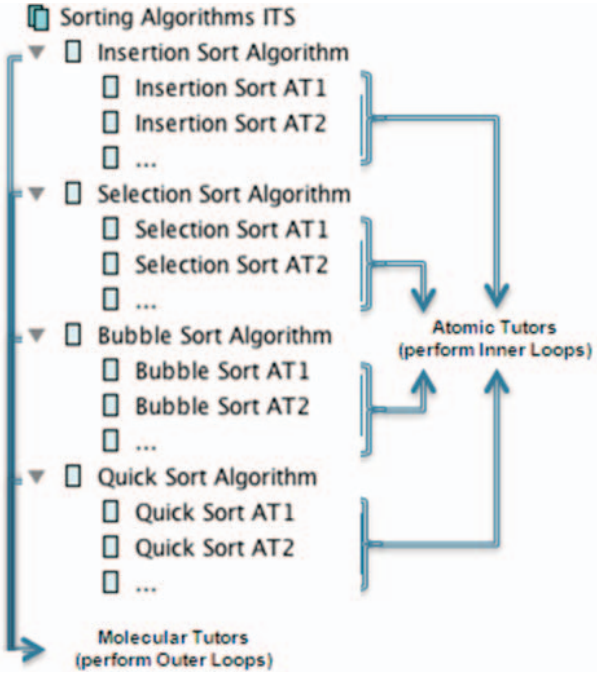

Fig. 2. Sample tree structure of an ITS for sorting algorithms.

are designed to perform task selection. Consequently, MTs implement outer loops, selecting the appropriate AT that should be delivered to a student at a specific moment. While ATs provide problem solving assistance and update the student model, MTs aggregate ATs and use the student model to select tasks.

This analogy with basic chemistry concepts appears appropriate because ATs might not be decomposed without losing functionality and also because they must be grouped to create MTs. A single MT is, in fact, a complete ITS for a specific topic. However, larger ITSs can also be composed by more than one MT (thus providing training for more than one subject, as shown in Fig. 1). Conceptually, an MT should represent an educational unit. Additionally, ATs should represent specific tasks for this unit.

For example, an MT could cover an educational topic such as "Calculating the Area of Triangles." Accordingly, the correspondent ATs would be activities and exercises for this topic. Furthermore, an MT for the domain of "Calculating the Area of Triangles" could be combined with several other MTs ("Calculating the Area of Squares," "Calculating the Area of Parallelograms," "Calculating the Area of Circles," and "Calculating the Area of Ellipses") to create an ITS for a larger educational unit, such as "Calculating the Area of Plane Shapes."

As another example of how to assemble ATs and MTs to create complete ITSs, let us consider the domain of "Sorting Algorithms." An ITS for this domain should be composed by MTs for the "Insertion Sort Algorithm," "Selection Sort," "Bubble Sort," and "Quick Sort." Each MT would have their own set of ATs for exercising their respective topics (see Fig. 2).

Our approach applies, therefore, a "divide-and-conquer" strategy, decomposing a large tutor into small tutoring artifacts that are grouped to create a complete ITS. The decomposition fits perfectly the paradigm of e-learning

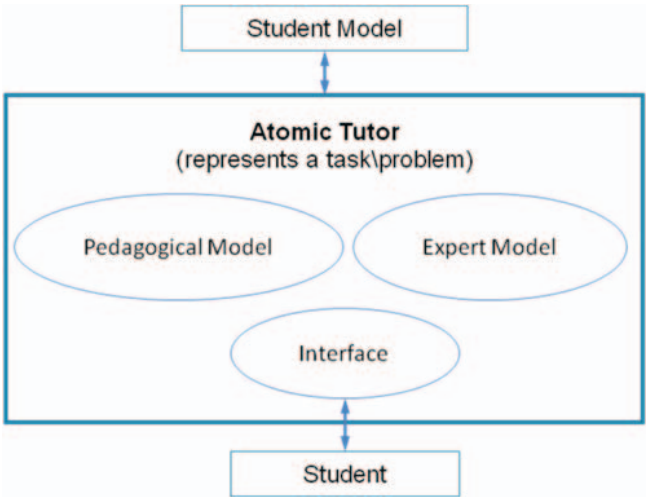

Fig. 3. Architecture of an AT.

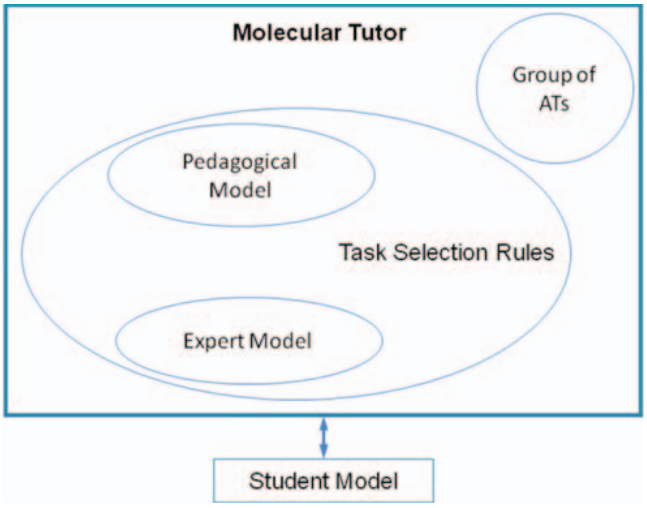

Fig. 4. Architecture of an MT.

standards because the standards are focused on the aggregation LOs. Some researchers in the field of ITSs have used a similar decomposition strategy to implement tutors with programming by demonstration [19], [20].

3.1 Architecture of Atomic ITSs

Because ATs resemble miniature tutoring systems for specific problems, in terms of architecture, they must have their own expert and pedagogical models, plus everything that they need to run coded in the same learning object (correct step representations, incorrect step representations, graphical widgets, error-specific feedback, hint sequences). From this point of view, ATs are very similar to example-tracing tutors [19], [20].

Fig. 3 shows that the expert model and the pedagogical model are coded together in the same LO. Some functions can execute instructions regarding the expert model and also instructions regarding the pedagogical model. Obviously, instructional designers can attempt to write the code in such a way that the pedagogical model is clearly separated from the expert model. However, the important idea to keep in mind is that ATs should behave according to the models that were previously defined, in spite of how the models are internally implemented.

Designing the models appropriately is clearly very significant for the pedagogical success of the ITS. Therefore, it is important to have some guidelines for reference. Cognitive task analysis (CTA) has been successfully used for designing ITSs for many years. Accordingly, running a CTA before starting the implementation of the ITS is strongly recommended because the task analysis will provide theoretical foundations for the functional design of the ITS and the ITS models.

3.2 Architecture of Molecular ITSs

Once MTs are responsible for aggregating ATs and performing task selection, their architecture has a major structural difference compared to ATs, which is the absence of a user interface. MTs do not necessarily need to interact with the students. They can, if interaction is desired by the instructional designers (for example, by using a table of contents). However, MTs must have a set of task selection rules, to consult the student model and to determine the most suitable activity.

Every time that a student starts a course or finishes working on an activity, MTs call the task selection rules and choose what AT should be delivered next. As Fig. 4 shows, to make instructional decisions, MTs have their own expert and pedagogical models coded in the task selection rules. Additionally, following the same principle used for ATs, the models should be designed after a CTA.

3.3 Implementation Details

To clarify how to practically implement our approach, the next sections will describe technical details about using SCORM (and associated technologies) to develop ITSs. Basically, for implementing the ITS loops, we rely on some SCORM constructs, especially the tracking data and the sequencing definition. The tracking data are used for registering the student model and the sequencing definition is used to perform task selection.

The ability to record information about students' performances in runtime in conjunction with the ability to use this information later for selecting activities are the two essential conditions for implementing ITSs. The SCORM sequencing and navigation loop is already designed from the beginning to allow task selection. Therefore, it naturally performs outer loops.

In contrast, implementing inner loops with SCORM activities is up to instructional designers. Basically, designers must be aware of the ITS services that they should implement and also how to update the student model using the SCORM tracking data. All of these goals can be accomplished in a perfecting manner using only standard web-development technologies.

3.3.1 Inner Loop Implementation

As described above, ATs are responsible for providing problem solving support and for implementing inner loop services (such as the assessment of knowledge, hints, and error-specific feedback). To help explain the inner loop implementation, let us first consider the architecture of an AT as presented in Section 3.1.

Code Excerpt 1. XHTML Interface Fragment.

```
...
<span id = "q1_wording"> −1) Angle 1 and 5 are?
</span> <select id = "question_1" onchange =
"question_1_rules()">
<option selected>.....</option>
<option>Alternate Interior Angles</option>
...
<option>Corresponding Angles</option>
</select>
<span id = "q1_img"></span>
...
<span id = "tutorStatus" class = "bolddText">Go ahead
and start solving the problem!</span>
...
```

To implement the models of an AT, the interface, the rules, and all of the requisites identified in the CTA, we use standard web-development technologies (as required by SCORM). These technologies include HTML, CSS, and ECMAScript. HTML and CSS are used only to build the interface. Everything else that requires processing information is coded in ECMAScript (more specifically, ECMAScript's most successful dialect, JavaScript). Code Excerpt 1 gives an example of how to process step-by-step problem solving.

Code Excerpt 2. Function "question_1_rules."

```
function question_1_rules(){
var grade = −1;
// Correct answer
if(document.getElementById("question_1").options[
document.getElementById
("question_1").selectedIndex].value
= = "Corresponding Angles") {
document.getElementById("question_1").disabled =
true;
document.getElementById("q1_img").innerHTML =
'<img src = "check.gif" alt = "This is correct!" />';
document.getElementById("tutorStatus").innerHTML =
"This is correct!";
grade = updateSkill(skill_1, 0.2);
} else{// Incorrect answer
question_1_buggy_answer();
grade = updateSkill(skill_1, −0.1);
}
myJsProgressBarHandler.setPercentage(skill_1 + '_bar',
Math.round(grade*100).toString());
}
```

The select clickable widget that we can see in Code Excerpt 1 is associated with a JavaScript event. When a student interacts with the drop-down list and performs a step of the problem, the function "question_1_rules" is called. This function will assess the step performed by the student, update the student model, and provide some feedback. Code Excerpt 2 presents some details relative to the JavaScript function "question_1_rules."

Because the step analyzed by the function above comprises selecting a single value from the drop-down list, it is the simplest example that we can have. It is very easy to verify the correctness of this step. Therefore, the function has only one (if, then, else) rule, which confirms whether the correct value was selected. If yes, then the AT provides positive minimal feedback, displaying a "this is correct message" and a green checkmark image. If the answer is wrong, then another function to diagnose the error will be called.

Diagnosing the error could result in other function calls, for example, to provide error-specific feedback or to display hints. Nevertheless, regardless of what answer is given by the student (correct or incorrect), the user model will be updated. In this specific case, if the answer is correct, then the proficiency of the skill "identifying correspondent angles" is increased by 20 percent. If not, the proficiency of the skill is decreased by 10 percent.

The method "myJsProgressBarHandler.setPercentage" reflects updating the user skills in the Interface "Skill-O-Meter." In contrast, updating the student model on the server database requires addressing the in-depth parts of the SCORM specification. This task includes working with the RTE API functions and also working with SCORM objectives.

Code Excerpt 3. Function "updateSkill."

```
function updateSkill(scormObjective, value) {
        var objectiveID = findObjective(scormObjec
        tive);
        var newGrade = getGrade(objectiveID);
        newGrade = newGrade + value;

        if(newGrade > = 1){
          newGrade = 1;
          doSetValue("cmi.objectives." + objectiveID +
          ".success_status", "passed");
        } else if(newGrade < 0) newGrade = 0;

        doSetValue("cmi.objectives." + objectiveID +
          ".score.scaled",
          (Math.round(newGrade*100)/100).toString());

        doCommit();
        return newGrade;
}
```

Objectives are special variables that are part of the tracking data and that can store some numerical values and statuses. Objectives can be used, for example, to represent grades or the proficiency of skills. When a student is working with an AT, the proficiency of the skills is recorded in the SCORM objectives using RTE functions. Code Excerpt 3 shows how to accomplish this task.

The function presented in Code Excerpt 3 receives two parameters. The first parameter is the name of the skill to be updated. The second parameter is the value of the update. Initially, a reference to the SCORM objective is retrieved using the name of the skill and the auxiliary function "findObjective." Then, the proficiency level of the skill is obtained using the auxiliary function "getGrade."

After obtaining the reference and the grade, the proficiency level of the skill is updated. Validation assures that the value stays between 0 and 100 percent. To conclude the updated process, the function "doSetValue" is called to

record the new value on the server and "doCommit" is called to commit the transaction. Both functions are part of the SCORM RTE API.

The example presented above does not involve a substantial amount of information processing. However, this simple instance fits the purpose of explaining the basic implementation mechanism. It is important to keep in mind that standard web-development technologies can be used for much more complex applications. Accordingly, everything that is necessary to assure the correct behavior of an AT can be implemented without any problem using JavaScript and associated web technologies.

Traditionally, inner loop services would be implemented using a rule-based language. However, they do not necessarily need to be designed using languages such as PROLOG or LISP. Inner loop services can actually be implemented in many different ways and still behave as rule-based systems [19], [20]. In contrast, it is important to remember that ATs should behave according to the models that were previously designed to be pedagogically successful. Once again, we stress the importance of performing a CTA before implementing an AT.

To summarize, for implementing the inner loop, HTML and CSS are used to characterize the interface. JavaScript is used to process information and handle events that produce changes to the interface. Finally, the SCORM RTE functions are used to handle the user model and to store and retrieve data about the students on the server.

3.3.2 Outer Loop Implementation

Because implementing outer loops is a responsibility of the MTs, let us initially review their architecture, as presented in Section 3.1.2. First, MTs aggregate ATs. Second, their main functionality is achieved by a set of task selection rules. These rules reflect the educational guidelines that were established in the expert and pedagogical models. Accordingly, the basic mechanism comprises using the rules to access the user model, which is stored in the SCORM objectives and subsequently using the student information to select tasks.

Code Excerpt 4. Outer Loop Code Fragment.

```
...
var indexOfIdentifyObjective1 = findObjective("Identify
Corresponding Angles");

grade1 = getGrade(indexOfIdentifyObjective1);
var indexOfIdentifyObjective2 = findObjective
("Calculate Corresponding Angles");
grade2 = getGrade(indexOfIdentifyObjective2);

var indexOfIdentifyObjective3 = findObjective("Identify
Vertical Angles");
grade3 = getGrade(indexOfIdentifyObjective3);

var indexOfIdentifyObjective4 = findObjective
("Calculate Vertical Angles");
grade4 = getGrade(indexOfIdentifyObjective4);
...
```

Basically, every time that it is necessary to select an activity, MTs call a special SCO to run the task selection rules. Therefore, this task selection SCO is part of the SCORM activity tree, as we can see in a sample SCORM-compliant ITS that is presented in Fig. 5. This SCO is launched whenever a course starts or whenever an activity is terminated. On the sample ITS above, the SCO is represented by the initial leaf in the activity tree.

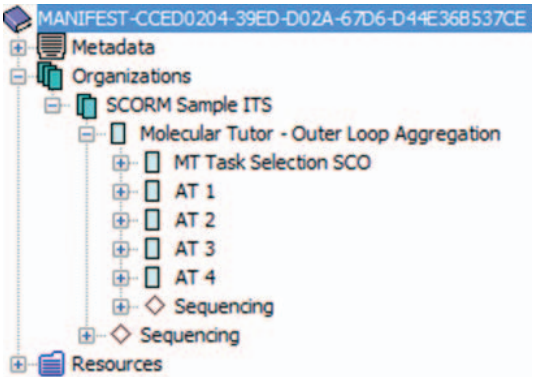

Fig. 5. SCORM activity tree of a sample ITS.

After the task selection SCO is called, the first thing to be accomplished is to access the values on the user model. As a result, this SCO references several objectives while loading, and the start of the task selection script contains instructions to retrieve the values of the SCORM objectives, as presented in Code Excerpt 4.

After loading the proficiency of the skills from the server database, a rule-based engine will operate on the data to determine the next suitable activity. The set of task selection rules of the engine have a similar structure to the code excerpt presented below:

Code Excerpt 5. Outer Loop Code Fragment.

```
...
if ((grade1 < X && grade2 < Y) || (grade1 < Z) || (grade2 <
W)){
        selectActivity("activity1");
} else if ((grade3 < X && grade4 < Y) || (grade3 < Z) ||
     (grade4 < W)){
        selectActivity("activity2");
} else if ((grade1 < A) || (grade2 < A)){
        selectActivity("activity3");
} else if ((grade3 < A) || (grade4 < B)){
        selectActivity("activity4");
}
...
```

The function "selectActivity," which is called in Code Excerpt 5, marks the desired SCORM activity as selected, writing a Boolean value on the activity's primary objective (every SCORM activity is associated with a primary objective). Activities that are not marked as selected are simply skipped by the SCORM sequencing mechanism. Therefore, only one AT will be loaded per iteration

When a student terminates working with an AT, one iteration finishes and another one starts. Accordingly, the task selection SCO is called again, objectives are retrieved from the database, and the rules will determine the next activity. This looping process on the activity tree is assured by the sequencing constructs of the SCORM specification (as shown in Fig. 5). For example, every SCORM activity has a

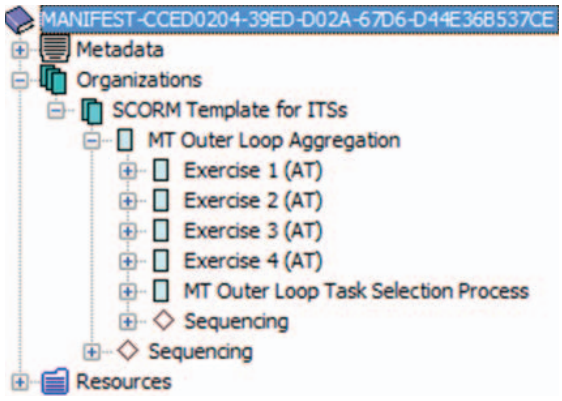

Fig. 6. SCORM activity tree of the implementation template.

precondition rule that can force an activity to be skipped. Additionally, MTs also have conditions on their SCORM sequencing definition to assure that the outer loop exits only when all of the instructions are considered to be completed.

SCORM provides a good built-in mechanism for implementing outer loops. However, implementing personalized task selection is never an easy task. First, it requires deep knowledge about the activities that are available and how to alternate between them. Second, it is necessary to be aware of all of the skills tracked in the user model, to determine the most suitable activity for a specific student.

3.3.3 Assembling Issues and Recommendations

Naturally, when instructors create courses using materials from different sources, they need to perform a careful analysis of the content to be included. Accordingly, assembling ITSs will always require an analysis of the tutors to be used.

In regard to assembling tutors developed by different parties, there is a danger of inconsistency among the models. However, because in our approach the ATs and MTs have their own models, it is reasonable to merge tutors that actually have distinct models. In fact, when it is necessary to merge ITSs, the reason is that they exercise different topics and different topics naturally have distinct models.

The main assembling issue is when the tutors to be aggregated exercise skills in common. When this situation occurs, it is necessary to ensure that the skill updates are coherent and do not interfere with each other. If instructional designers model the students by estimating the proficiency of the skills, which we recommend in our approach, the assembling process will be easier. After all, "the student model is nothing more than a fancy grade book" [20]. Accordingly, any aggregating LOs that is composed of more than one ITS can simply access the proficiency values, or "grades," that are stored in the user model and make decisions.

Finally, there are many cases for which our LOs can be used without being assembled into larger tutors. For example, they can be used for creating courses that simply add the LOs to a course syllabus (similar to creating a course in Moodle). Creating an aggregating LO that contains a course syllabus allows the component ITSs to work apart from each other. In many cases, this structure should be sufficient to create good courses that are characterized by tutors that work independently.

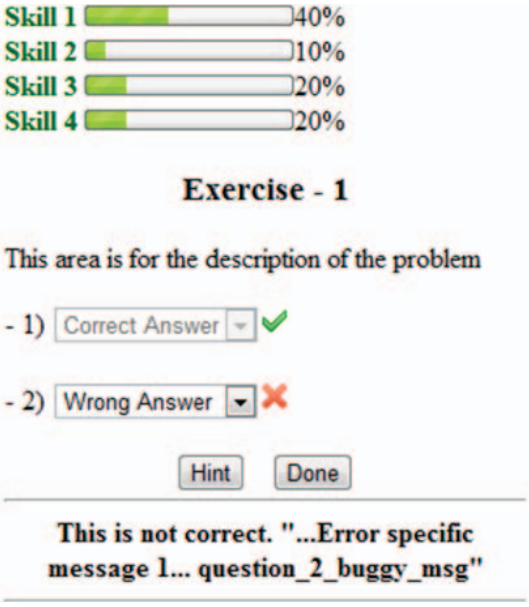

Fig. 7. Implementation template interface snapshot.

3.3.4 Implementation Template

Once our approach is open source and also developed to be used by others, an implementation template was created. Developing SCORM templates is a common practice within the community working with these standards. Accordingly, our template was designed to assist others in using our implementation method. In addition, the SCORM PIF that composes the template can be very useful for understanding in practice how the approach works, which complements the information in this paper.

The template was developed to be as simple as possible, but to use all of the attributes of the approach, having examples of everything that it is necessary to implement an ITS. As the SCORM activity tree in Fig. 6 shows, the template has one MT and four ATs.

When loaded, the application launches the first task on the activity tree, which is Exercise 1. As a result, the educational platform will display the associated LO, as shown in Fig. 7. The interface of this LO has a skillometer, space for characterizing the problem (with anything judged to be appropriate, such as images or the wording of the problem) and a form with two clickable widgets that represent two steps.

All of the ATs that compose the template have the same aspect, but they are linked to different skills in the user model. As shown by the skillometer, this dummy ITS handles four skills and the outer loop task selection rules will alternate between the ATs based on the proficiency of these skills.

Everything on the template works in a manner similar to a real ITS (e.g., multilevel hints, error specific feedback sequences, minimal feedback, task selection). However, users obtain only simulated feedback messages because the template implements only the code for the skeleton of the functionalities; no meaningful content is displayed. Naturally, the template can be simply edited to produce pedagogically sound educational software.

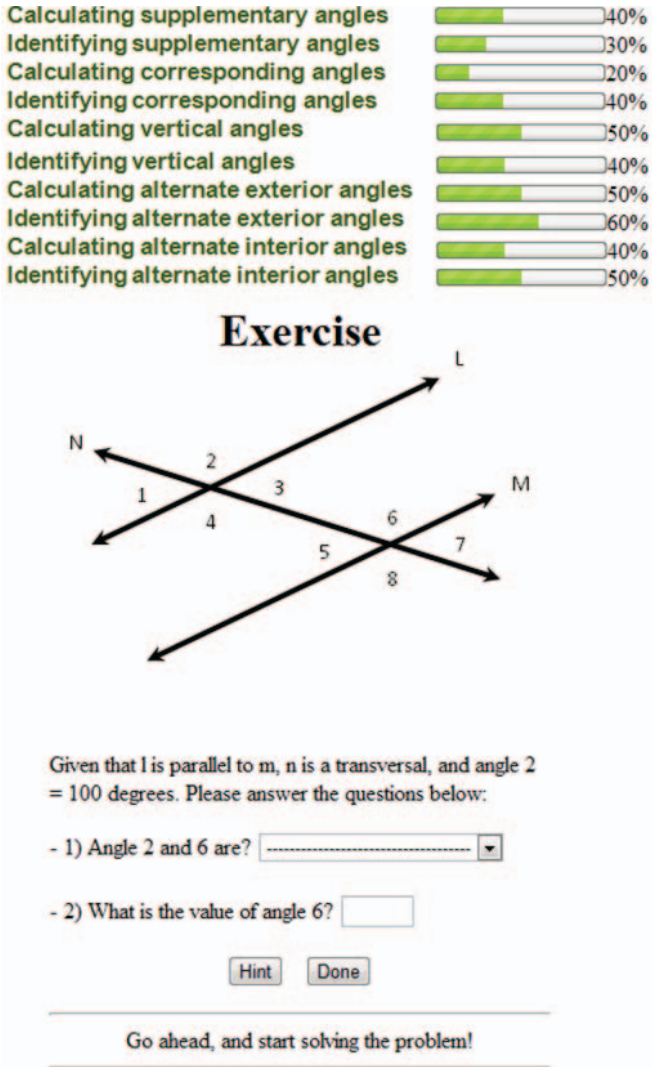

Fig. 8. Prototype interface snapshot.

4 ITS Prototype

To validate our approach, a small SCORM-compliant ITS was developed. The implementation tools were Dreamweaver IDE (for web development) and RELOAD IDE (for the SCORM PIF). The domain of the application is middle school math, and the tutor is composed of one MT and five ATs.

Each AT is designed to exercise different skills about "angles formed by parallel lines." Accordingly, the skills involved in our prototype are: calculating supplementary angles, identifying supplementary angles, calculating corresponding angles, identifying corresponding angles, calculating vertical angles, identifying vertical angles, calculating alternate interior angles, identifying alternate interior angles, calculating alternate exterior angles, and identifying alternate exterior angles.

The method for grading skills on this prototype uses a range from 0 to 100 percent. Accordingly, every step necessary to solve a problem is mapped to a corresponding skill. Skills are initiated with a value of 20 percent. When a student answers a step correctly, the value of the corresponding skill is increased by 20 percent. If a student answers a step incorrectly, then the value of the corresponding skill is decreased by 10 percent. When a student asks for a hint, then the value of the corresponding skill is decreased by 5 percent.

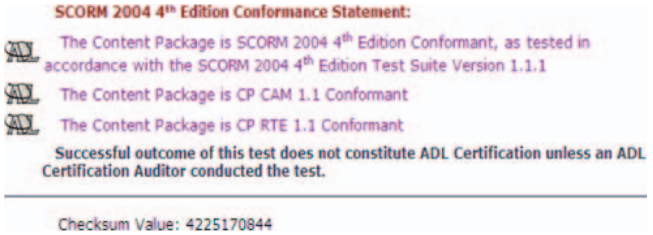

Fig. 9. ADL SCORM conformance test.

The prototype suggests exercises and provides hints, minimal feedback, error-specific feedback, and assessments of knowledge. The current version of our user interface is composed of a skillometer, an image, the wording of the problem, and a form (as shown in Fig. 8). Our prototype was recently described in detail in another article. For further details about the application, consult a book chapter that was recently published by the authors [10].

5 Approach Evaluation

To evaluate our approach for implementing interoperable ITSs, the first step was to submit our prototype to a SCORM conformance test using the ADL SCORM test suite. Verifying compliance is an important step because it guarantees the correctness of the SCORM package. Accordingly, the ADL test suite uses a step-by-step process to validate the whole SCORM application, including the required API calls of each SCO.

Fig. 9 shows a snapshot with the results of the conformance test of our prototype. This figure shows that the package is compliant with all of the ADL requirements, and therefore, the SCORM PIF should run correctly, assuring the interoperability of the educational software. However, to guarantee that everything works appropriately, we have tested our prototype in different educational platforms, using different browsers and also different operating systems (see Figs. 10, 11, 12, and 13). The tests were conducted to verify the functionalities of the inner loop and the functionalities of the outer loop. Accordingly, the basic procedure that was used for testing was the following:

1. Access the SCORM compliant educational platform and import the ITS.
2. Check for import errors or warnings.
3. Load the ITS.
4. Check for loading errors or warnings.
5. Verify if the outer loop selected the correct problem.
6. Solve the problem to verify the inner loop functionalities, step by step, one by one.
7. Repeat Steps 5 and 6 until instruction is complete.

The platforms used for testing were the SCORM sample runtime environment (SRTE), the Odijoo LMS, the SCORM cloud educational system, and Moodle. Each educational platform was tested using different navigators, except for the SCORM SRTE, which runs only on Internet Explorer. Each browser ran on a different operating system. For example, Internet Explorer ran on Windows XP, Chrome ran on Windows 7, and Safari ran on Mac OS X.

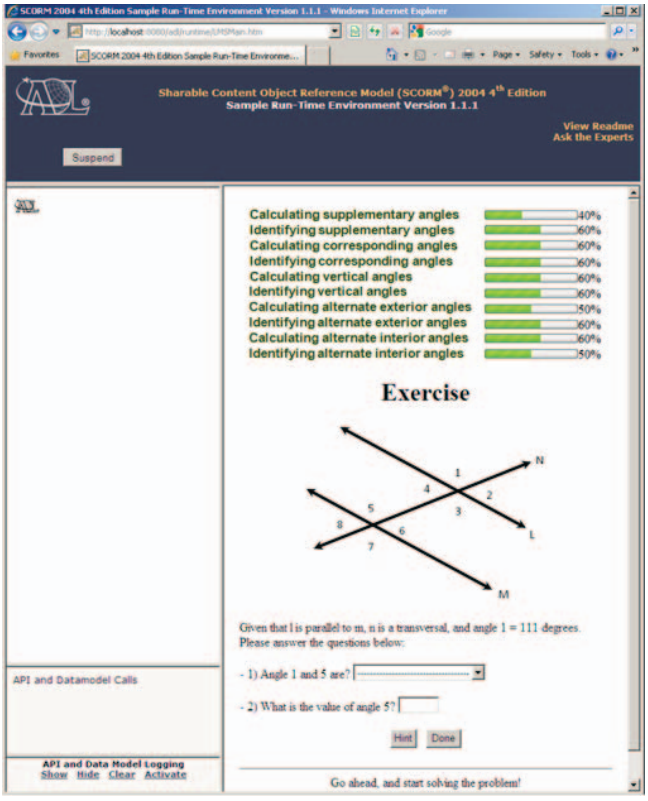

Fig. 10. Prototype loaded on the SRTE.

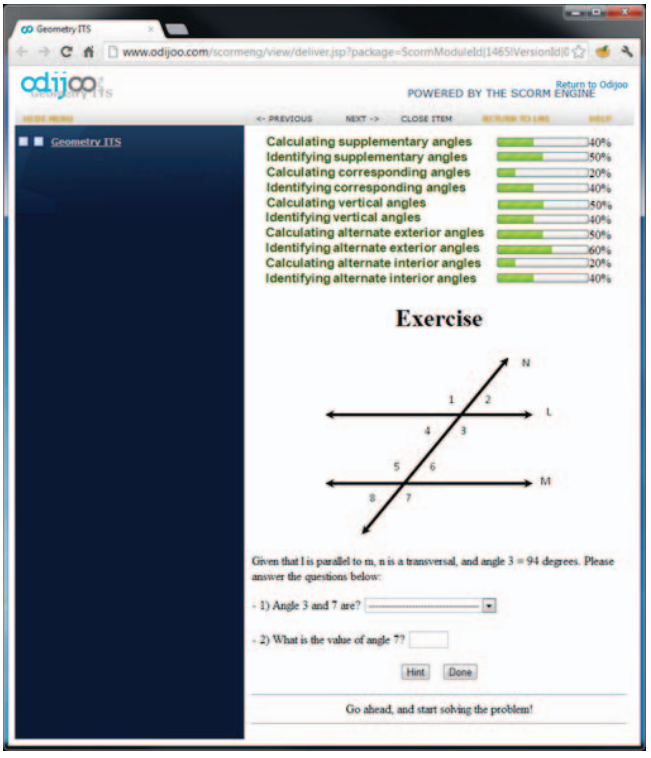

Fig. 11. Prototype loaded on Odijoo.

The SCORM SRTE, the Odijoo LMS, and the SCORM Cloud educational platforms can naturally support SCORM 1.3. Accordingly, they do not need any extensions to run the tests. In contrast, Moodle supports only part of the SCORM 1.3 specification (sequencing and navigation is still not supported). Consequently, Moodle requires the installation of an external module to run applications that use sequencing and navigation constructs (in our case, we used the Rustici SCORM cloud Moodle Mod).

Fig. 10 shows the prototype loaded in the SRTE, using Internet Explorer on Windows XP. Fig. 11 shows the prototype loaded in the Odijoo LMS, using Chrome on Windows 7. Fig. 12 shows the prototype loaded in the SCORM cloud, using Safari on Mac OS X. Fig. 13 shows the course statistics of the ITS in Moodle, after instruction is complete.

To summarize the results of the experiments, we present the tables that regard the functionality tests on each educational system. The tests showed that our prototype can run correctly in SCORM-compliant educational platforms. Table 1 shows that our method can support the functionalities of the inner loop and the functionalities of the outer loop. Therefore, we have confirmed that our approach can be successfully used to implement interoperable tutors.

If we compare our approach for interoperability with other approaches for delivering ITSs to LMSs, the main advantage of our method is that we can build web-based ITSs that not only can be loaded to some LMSs but also are fully compliant to standards, interoperable and open source. Table 2 presents a short comparison of our technique with other methods. This table shows that all of the approaches can support the implementation of adaptive educational systems that can be integrated with LMSs. However, our approach is the only approach that can assure the interoperability of the LOs because of the usage of standard technologies and the absence of significant external dependencies. In addition, our approach can also produce tutors that are OERs, which results from the open-source features of our method.

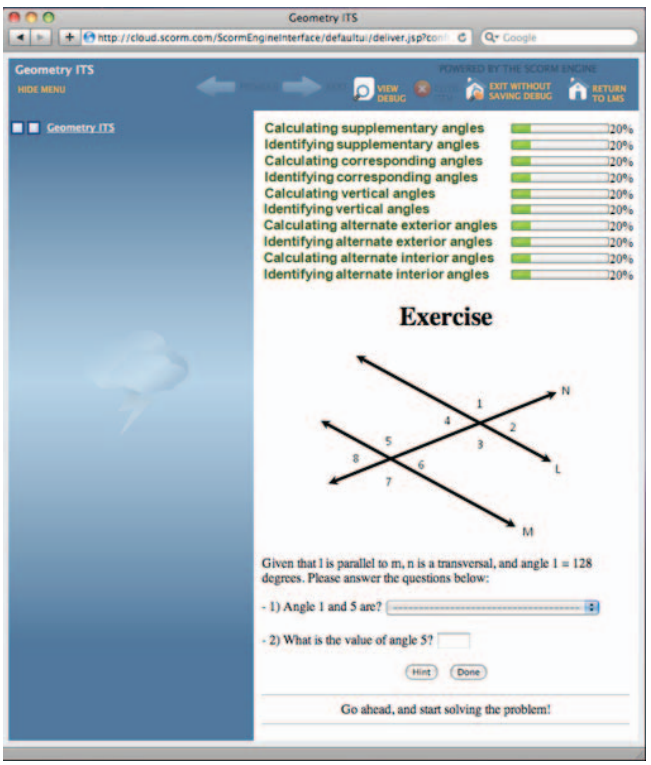

Fig. 12. Prototype loaded on the SCORM cloud.

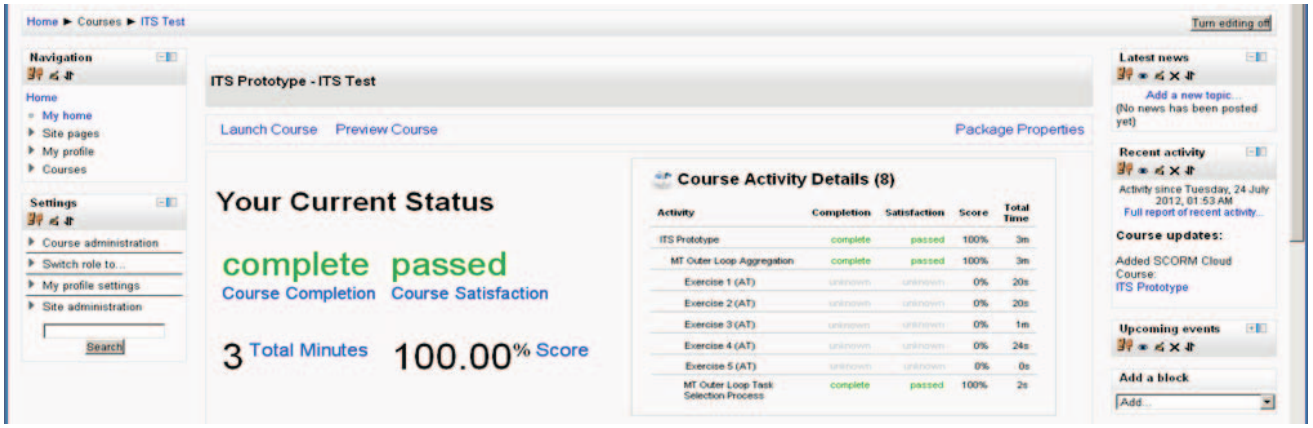

Fig. 13. Course status of the prototype loaded on Moodle.

TABLE 1
Functionality Tests

	Educational Platform	Moodle			SCORM Cloud			Odijoo			SCORM SRTE
	Browser	Chrome	Safari	Explorer	Chrome	Safari	Explorer	Chrome	Safari	Explorer	Explorer
	ITS Functionalities										
Inner Loop	Step Based Problem Solving	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Minimal Feedback	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Error Specific Feedback	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Multilevel Hints	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Review of Solutions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Read From the User Model	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Write on the User Model	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Outer Loop	Read From the User Model	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Write on the User Model	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Obtain a Rich Set of Tasks	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Select Tasks	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Deliver Tasks to the User	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE 2
Comparison of Approaches

	Features	GRAPPLE Aproach	T-Maestro Aproach	Our Aproach
Suports	Web-Based LOs	✓	✓	✓
	Personalization and Adaptation	✓	✓	✓
	Integration with LMSs	✓	✓	✓
	LOs Fully Compliant to Standards	✗	✗	✓
	Interoperable LOs Free of Significant External Dependencies	✗	✗	✓
	Open Source and Reusable Implementation Code	✗	✗	✓
Provides	Authoring Tools	✓	✓	✗
	Open Source Implementation Template	✗	✗	✓

Regarding implementation tools, we recognize the importance of having an authoring framework to assist in the development process. However, because of the limited human resources that were assigned to our project at this stage, authoring tools might be developed only in a future phase, especially because they have no influence on our conceptual framework for interoperable tutors. Nevertheless, in spite of the small size of our team, and to assist in using our method, we have already developed an open-source implementation template that supplies all of the examples and built-in structures that are necessary to design ITSs using our approach.

GRAPPLE and T-Maestro perform a high-quality job with respect to providing authoring tools for developing adaptive web-based educational systems. These projects also provide some advanced services for supporting sophisticated student models. Examples include the GRAPPLE user modeling framework (GUMF) and the T-Maestro user profile manager. Both projects allow the development of educational systems that use refined personalization techniques. However, these approaches significantly use nonstandard technologies to support adaptation. In fact, most of the procedures necessary to perform user modeling and task selection run on nonstandard external frameworks, which is not necessary in our approach.

Using external and nonstandard technologies for implementing interoperable ITSs can be considered, therefore, redundant. However, using current technology, the

dependence on external services for implementing adaptation has a few tradeoffs. First, using specialized services such as GUMF and the T-Maestro user profile manager have the advantage of supporting cutting edge student models and especially flexible adaptive applications. In contrast, with regard to the interoperability and reusability of LOs, there are several issues.

Essentially, interoperability is the main purpose of standardization. Accordingly, if an LO is considerably dependent on nonstandard external resources, then interoperability cannot be guaranteed. In addition, tying LOs to peripheral systems that process key features of the application can compromise the interoperability of the resources because some problems might arise because of the significant reliance on the other systems. For example, routine tasks such as system maintenance, adding user accounts, managing concurrent accesses, managing scalability, and many others might be out of the control of the stakeholders. Furthermore, if an external system is discontinued, updated, or simply crashes, dependent applications could stop working.

Another issue about using external systems to process key information is that they might preclude access to important parts of the source code. Therefore, updates and improvements in the educational software are limited to the available parts of the code, which is not desirable for OERs. Moreover, storing student data (such as grades, profiles, and cognitive models) in third-party frameworks could raise questions that are related to privacy and human subject research. For example: Who owns the data collected from the LOs? Can they be used for further studies without permission of the original author of the LO? Who can have access to the collected data? These questions can be especially relevant if the LOs are used not only for experimentation and testing but also in real-world educational settings, including, for example, student grades.

The main advantage of our method is that it uses only standard technologies for implementing ITSs as OERs. Therefore, our approach can guarantee the interoperability of the LOs and can additionally assure the open-source distribution of resources. For example, if an institution adopts a new platform or if significant updates of the platform are performed, the LOs being used by the institution will continue to work. Furthermore, because of our special architecture and also because of the usage of standards, it is possible to combine and group ITSs that are built with our method, which can be problematic using nonstandard technologies.

6 Conclusions and Future Work

This paper describes an approach for the development of interoperable ITSs using e-learning standards. Our approach is based on the development of atomic tutoring systems that are grouped to create molecular tutors, covering the curriculum of courses. In addition, our approach focuses on assuring what defines a tutor in terms of behavior and functionalities (inner loops and outer loops).

In contrast to other approaches, our method does not require extending standards with nonstandardized peripheral systems or databases (which restricts the interoperability of the LOs and the reusability of code). Therefore, our technique has the advantage of producing standardized tutors that are completely free of external resources and open source. As a result, our method allows implementing ITSs that can be downloaded from repositories of learning objects, loaded into different educational platforms, accessed over the web, combined with other LOs and have their open source code shared by communities, similar to any other OER.

For future work, we are following the release of the recently announced SCORM Tin Can API. This new API is currently being announced as the first step toward the new version of SCORM. Tin Can is promising more powerful ways of storing data about the users and groups of users. In this way, we expect to be able to record additional data about students, use more sophisticated methods, and support collaborative learning in the future.

Acknowledgments

This work was partially supported by the Portuguese Foundation for Science and Technology through individual grant SFRH/BD/66225/2009 and pluriannual INESC-ID research grant project PEst-OE/EEI/LA0021/2011.

References

[1] B. Beatty and C. Ulasewicz, "Faculty Perspectives on Moving from Blackboard to the Moodle Learning Management System," *TechTrends*, vol. 50, pp. 36-45, 2006.

[2] A.A. Piña, "An Overview of Learning Management Systems," *Learning Management System Technologies and Software Solutions for Online Teaching: Tools and Applications*, Y. Kats, ed., pp. 1-19, Information Science Reference, 2010.

[3] K. Sabbir Ahmed, "A Conceptual Framework for Web-Based Intelligent Learning Environments Using SCORM-2004," *Proc. IEEE Int'l Conf. Advanced Learning Technologies*, pp. 12-15, 2004.

[4] A.T. Corbett and J.R. Anderson, "The LISP Intelligent Tutoring System: Research in Skill Acquisition," *Computer Assisted Instruction and Intelligent Tutoring Systems: Establishing Communication and Collaboration*, J. Larkin and R. Chabay, eds., Erlbaum, 1992.

[5] S. Ritter, J.R. Anderson, K.R. Koedinger, and A. Corbett, "Cognitive Tutor: Applied Research in Mathematics Education," *Psychonomic Bull. Rev.*, vol. 14, pp. 249-255, Apr. 2007.

[6] K. Vanlehn, C. Lynch, K. Schulze, J. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill, "The Andes Physics Tutoring System: Five Years of Evaluations," *Proc. 12th Int'l Conf. Artificial Intelligence in Education*, pp. 678-685, 2005.

[7] J.E. McCarthy, "Military Applications of Adaptive Training Technology," *Technology Enhanced Learning: Best Practices*, D.G. Lytras, P. Ordóñez de Pablos, and W. Huang, eds., pp. 304-347, IGI Global, 2008.

[8] G. Santos and A. Figueira, "Web-Based Intelligent Tutoring Systems Using the SCORM 2004 Specification—A Conceptual Framework for Implementing SCORM Compliant Intelligent Web-Based Learning Environments," *Proc. IEEE Int'l Conf. Advanced Learning Technologies (ICALT '10)*, pp. 676-678, 2010.

[9] G. Santos and A. Figueira, "Reusable and Inter-Operable Web-Based Intelligent Tutoring Systems Using SCORM 2004," *Proc. Ninth European Conf. E-Learning*, 2010.

[10] G. Santos and J. Jorge, "Interoperable Intelligent Tutoring Systems as SCORM Learning Objects," *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends*, Peña-Ayala, ed., pp. 133-160, Springer-Verlag, 2012.

[11] G. Santos and J. Jorge, "Atomic and Molecular Intelligent Tutoring Systems—A New Architecture for Interoperable Tutors as Open Educational Resources," *Proc. IEEE Int'l Conf. Advanced Learning Technologies (ICALT '13)*, 2013.

[12] ADL, "SCORM," http://www.adlnet.gov/Technologies/scorm, 2011.

[13] ADL, "Who We Are," http://www.adlnet.gov/About/Pages/Default.aspx, 2011.

[14] ADL, "SCORM Benefits," http://www.adlnet.gov/Documents/SCORM%20FAQ.aspx#scorm, 2011.
[15] K.R. Koedinger and A.T. Corbett, "Cognitive Tutors: Technology Bringing Learning Science to the Classroom," *The Cambridge Handbook of the Learning Sciences,* R.K. Sawyer, ed., pp. 61-78, Cambridge Univ. Press, 2006.
[16] P. Brusilovsky, "The Construction and Application of Student Models in Intelligent Tutoring Systems," *J. Computer and Systems Sciences Int'l,* vol. 32, no. 1, pp. 70-89, 1994.
[17] A.T. Corbett, K.R. Koedinger, and J.R. Anderson, "Intelligent Tutoring Systems," *Handbook of Human-Computer Interaction,* second ed., M.G. Helander, T.K. Landauer, and P.V. Prabhu, eds., Elsevier, 1997.
[18] M.C. Polson and J.J. Richardson, *Foundations of Intelligent Tutoring Systems.* Lawrence Erlbaum, 1988.
[19] V. Aleven, B.M. McLaren, and J. Sewall, "Scaling Up Programming by Demonstration for Intelligent Tutoring Systems Development: An Open-Access Web Site for Middle School Mathematics Learning," *IEEE Trans. Learning Technologies,* vol. 2, no. 2, pp. 64-78, Apr.-June 2009.
[20] V. Aleven, B.M. McLaren, J. Sewall, and K.R. Koedinger, "A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors," *Int'l J. Artificial Intelligence in Education,* vol. 19, pp. 105-154, 2009.
[21] K. Vanlehn, "The Behavior of Tutoring Systems," *Int'l J. Artificial Intelligence in Education,* vol. 16, pp. 227-265, 2006.
[22] P. De Bra, D. Smits, K. Van der Sluijs, A. Cristea, J. Foss, and C. Steiner, "GRAPPLE: Learning Management Systems Meet Adaptive Learning Environments," *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends,* A. Peña-Ayala, ed., vol. 17, pp. 133-160, Springer-Verlag, 2012.
[23] M. Rey-López, P. Brusilovsky, M. Meccawy, R. Díaz-Redondo, A. Fernández-Vilas, and H. Ashman, "Resolving the Problem of Intelligent Learning Content in Learning Management Systems," *Int'l J. E-Learning,* vol. 7, pp. 363-381, 2008.
[24] M. Rey-López, R. Díaz-Redondo, A. Fernández-Vilas, J. Pazos-Arias, J. García-Duque, A. Gil-Solla, and M. Ramos-Cabrer, "An Extension to the ADL SCORM Standard to Support Adaptivity: The T-Learning Case-Study," *Computer Standards & Interfaces,* vol. 31, pp. 309-318, 2009.
[25] P.D. Bra and D. Smits, "A Fully Generic Approach for Realizing the Adaptive Web," *Proc. 38th Int'l Conf. Current Trends in Theory and Practice of Computer Science,* 2012.

Gustavo Soares Santos received a degree in information science engineering from the Polytechnic of Porto, in 2006, where he was a lecturer for two years. He is currently working toward the PhD degree in computer science at the Instituto Superior Técnico, Technical University of Lisbon. He participated in the CMU-Portugal PhD program and conducted part of his PhD studies at Carnegie Mellon University. His research interests include intelligent tutoring systems, adaptive and personalized instruction, web-based learning, e-learning standards, and human-computer interaction.

Joaquim Jorge coordinates the VIMMI research group at INESC-ID and is a full professor at the Instituto Superior Técnico, Technical University of Lisbon. He is the editor-in-chief of the *Computers and Graphics Journal* (Elsevier). He helped organize 35 scientific events including ACM Intelligent User Interfaces 2012 (IPC cochair) and INTERACT 2011 (cochair), served on 150 international program committes, and coauthored more than 225 publications in international refereed journals, conferences, and books. His interests include distance learning, interactive computer graphics, and multimodal user interfaces. He is a fellow of the Eurographics Association and a senior member of the ACM and the IEEE Computer Society.