International Jl. on E-Learning (2008) **7***(3),* 363-381

Resolving the Problem of Intelligent Learning Content in Learning Management Systems

MARTA REY-LÓPEZ
University of Vigo, Spain
mrey@det.uvigo.es

PETER BRUSILOVSKY
University of Pittsburg, USA
peterb@pitt.edu

MARAM MECCAWY
University of Nottingham, UK
mxm@cs.nott.ac.uk

REBECA DÍAZ-REDONDO AND ANA FERNÁDEZ-VILAS
University of Vigo, Spain
rebeca@det.uvigo.es
avilas@det.uvigo.es

HELEN ASHMAN
University of South Australia, Australia
Helen.Ashman@unisa.edu.auv

Current e-learning standardization initiatives have put much effort into easing interoperability between systems and the reusability of contents. For this to be possible, one of the most relevant areas is the definition of a run-time environment, which allows Learning Management Systems to launch, track and communicate with learning objects. However, when dealing with intelligent content, these environments show important restrictions. In this article, we study these restrictions, comparing several standardized run-time environments with nonstandardized solutions that aim to overcome these constraints.

Modern approaches to e-learning are based on the reusability of learning content. The core idea is simple: once quality learning content is created, it should be reused multiple times in different contexts. To make this simple idea work, a number of interoperability solutions must be provided. Each piece of the content should be properly stored in a repository and tagged with

critical metadata that will make it retrievable. Tools must be provided to teachers so that they can locate relevant content and include it into their courses. Finally, a Learning Management System (LMS) should be able to deliver this content properly, even when the content has not been specifically designed to run with the chosen LMS. Work on reusability was begun by both academia and industry and, after several years of research and organizational work, diverse e-learning standards have been created by several bodies – both public and private – such as Institute of Electrical and Electronics Engineers (IEEE) Learning Technologies Standardization Committee[1] (LTSC), Institute of Mathematical Statistics (IMS) Global Learning Consortium[2], Aviation Industry Computer-Based Training (CBT) Committee[3] (AICC) or Advanced Distributed Learning[4] (ADL). These standards focused mainly on the reusability of content and interoperability between systems.

These sets of standards answered most of the needs of those authors who operated mainly with straightforward content and courses – web pages, lecture slides, and small multimedia items. This content can be easily stored, retrieved, packaged, moved to an LMS and added to course pages. However, this is not the kind of content that provides the most solid pedagogical impact. It has been observed multiple times that an LMS stuffed with simple content is just a new reincarnation of the *page turning* approach that has long been criticized as a poor way to learn. Since the first attempts to automate *page turning* in classic Computer Assisted Instruction systems in 1970, research on the use of computers in education has suggested different kinds of so-called *intelligent content*. The idea of intelligent content is to engage a student in a meaningful learning activity – such as exploration or problem solving – and to provide him/her with the necessary support.

The most popular examples of rich content are simulations and Intelligent Tutoring Systems. Simulations (frequently augmented with advanced visalization) engage the student in exploration of complicated concepts and help to uncover the dynamics of important processes. Intelligent Tutoring Systems assist the student in solving challenging problems, providing help on each step towards the solution or deep analysis of solution errors. Both technologies have been used to enhance the teaching of various subjects ranging from physics (McKenna & Agogino, 1997; VanLehn et al., 2005) to Computer Science (Brusilovsky, Schwarz, & Weber, 1996; Butler & Brockman, 2001).

The problem is that existing reusability solutions do not provide appropriate support for using intelligent content in modern LMSs, due to several obstacles. For example, a piece of rich content is typically not a file. These activities cannot be simply packaged, stored or copied. For example, Episodic Learner Model-Adaptive Remote Tutor (ELM-ART), an adaptive List Processing Language (LISP) course (Brusilovsky et al., 1996), includes many LISP programming problems, fully interactive learning activities backed by ELM-ART's unique knowledge-based functionality. In response

to a student program solution sent to the ELM-ART server, the system can check, diagnose, and correct it. However, ELM-ART exercises cannot be moved or copied – they must be served directly from a dedicated ELM-ART server (Brusilovsky, 2004).

This article focuses on another obstacle that prevents the use of intelligent content: the lack of support for collecting, processing, and using the results of the user's work with this content. A typical learning activity engages the student for some reasonable period of time, while the student performs many pedagogically important actions, like changing simulation parameters or making a step towards a problem solution. Many intelligent systems register these actions, to store the history of the student's work and build a student model, which can be used to provide personalized support for him/her. Access to the trace of user actions and to the student model produced by processing it is important for almost all the *stakeholders* of the e-learning process. The LMS needs it to recognize the completion of educational objectives and to organize the sequencing of learning content. The original system that produced this data needs to trace progress to provide continuity and proper adaptation. Other types of intelligent content used by the same student may also need to be notified of any updated state of the student goals and knowledge observed by an activity for them to provide adequate adaptation. The teacher may want to monitor the student's work with an activity to recognize problems and to assist him/her. The students may wish to observe the growth of their knowledge over the course of various learning activities. Overall, the ability to access and use this information is one of the keys to the success of intelligent content in modern e-learning.

This problem has been recognized by both researchers and practitioners. A few existing standards have attempted to solve the problem of communicating information between an LMS and a piece of intelligent content. However, these solutions have been geared toward simple kinds of rich content (such as questions, Flash animations, etc.) and have failed to resolve the problem. In turn, several groups of researchers working on intelligent and personalized content have suggested alternative interoperability frameworks to better answer the needs of personalization and broader access to this information.

The goal of this article is to analyze the problem of sharing the information about the student work with rich content and the processed results of its work in the context of modern e-learning. First, we examine the needs of intelligent content. Then, we provide an analysis of two standardized solutions: the ADL SCORM Run-Time Environment and IMS Shareable State Persistence. The objective is to identify the benefits and deficiencies of these approaches so that they can be addressed in further work. To demonstrate possible ways to resolve the interoperability problems, we present several nonstandardized solutions (Advanced Distributed Architecture for Personalized Teaching and Training [ADAPT^2], Methodology and Tools for the

Development of Intelligent Teaching and Learning Environments [MEDEA] and T-Learning Multimedia Adaptive Educational SysTem Based on Reassembling TV Objects [T-MAESTRO]). It is our hope that this article will be useful for all developers and users of intelligent content and will help the domain to move closer to the ultimate goal of this work: establishing a framework that supports a broad use of intelligent learning objects in modern e-learning systems.

THE NEEDS OF INTELLIGENT CONTENT

The use of simple content in an LMS is rather straightforward. A piece of presentation content is stored in the LMS, invoked by a link, and the system immediately records that it was presented.

The use of intelligent content is a more complicated procedure. It needs to have the user authenticated to attribute the result of the work to this user and adapt further instruction to him/her. After authentication, but before the very start of the work, intelligent content may need to have access to the history of the user's work in that subject area and to the user model. The former is required to provide continuity in complex multi-step activities; the latter, to adapt to the user – that is, take into account the starting level of user knowledge. The results of working with the content have to be reported to the rest of the world so that it can be considered by other stakeholders in the e-learning process. It can be done in the form of a trace, a student model, or both. While working with the user, the system may store the trace of student actions in the student record, which is accessible to those who need it. At the same time (or instead of it), the system may build its own model of user knowledge and interests. Work with the activity may be ended in several ways: the student can complete it, leave it or simply switch to another piece of content without properly abandoning the current one. Ending work with the activity should be registered either by the LMS or the content and communicated to the other components that need to know about it. In addition, when work stops, the student model accumulated by the intelligent content should be passed to the LMS as well as to other components that may need it. Thus, just one session of work with intelligent content requires a range of information exchanges where different information is communicated between diverse components.

The systems that we analyze provide different answers to this communication need. Examining these differences is important to determine *the right way* to solve the problem. To make our presentation more structured, we will attempt to focus on the same set of issues each time:

- How is the user being authenticated?
- How is the information about the user being passed to intelligent content?
- How can the trace of the user' work be stored?

- How can the end of work with the content be registered?
- How can the student model, accumulated by intelligent content, be shared with other components?

STANDARDS

To achieve reusability and interoperability, one of the key components of e-learning standards is the run-time environment. Its basic tasks are the delivery of content to the student, the support of the interaction between the content and the LMS and the decision of which content should be delivered next[5].

This work was pioneered by AICC and published in Computer Managed Instruction Guidelines for Interoperability (AICC, 2004), a technical report which defines (a) a general way to begin structuring learning contents, (b) a common mechanism for them to communicate with a Computer Managed Instruction system, and (c) a predefined data model for this communication. This technical report has been submitted to the IEEE LTSC and adopted as a standard for the latter (IEEE LTSC, 2004; IEEE LTSC, 2005). The run-time environment of the ADL Sharable Content Object Reference Model (SCORM) 2004[6] includes an adapted version of the IEEE LTSC standards aforementioned. In the next sections, we study ADL SCORM run-time environment, focusing on its interoperability with intelligent content, as well as Information Management Systems (IMS) Sharable State Persistence, an extension proposed to partially solve current SCORM restrictions in this matter.

ADL SCORM

The SCORM standard references specifications, standards, and guidelines developed by other organizations that have been adapted and integrated with one another to form a more complete and easier-to-implement model. It is a collection of specifications and standards that have been bundled into a set of technical books: Content Aggregation Model, Sequencing and Navigation, and Run-time Environment (RTE). In this section, we study the most relevant characteristics of the latter, focusing on its capabilities to launch and track intelligent content, which is called Sharable Content Objects (SCOs)[7] in SCORM terminology.

The SCORM RTE defines: (a) the Launch process as a common way for LMSs to start web-based content objects; (b) the Application Programming Interface (API) as the mechanism for exchanging data between the LMS and the SCO; and (c) a Data Model as a standard set of elements to define the information tracked for a SCO.

Concerning the *Launch process*, the LMS is responsible for determining the fully qualified URI of the content object, launching the Assets (content objects that are not able to communicate with the LMS) using the HyperText

Transfer Protocol (HTTP) protocol, as well as launching the SCOs (only one per learner at a time). The SCOs must be started in a web browser window that is dependent or a child frame of the window that contains the communication API, so that the SCO can access it by performing a recursive search.

In relation to the *API*, all the communication must be initiated by a SCO. SCORM RTE defines three different categories of methods: (a) *session methods*, which indicate the beginning and end of a communication session between the SCO and the LMS; (b) *data-transfer methods*, which allow the SCO to read/write values for elements in the data model; and (c) *support methods*, which are used for auxiliary communications, such as error handling. The SCOs are only required to call these methods when they begin and end a session with the LMS.

The purpose of establishing a *Data Model* is to standardize information about SCOs that can be tracked by different LMSs. It contains data about the status of the SCO, score data and thresholds for passing scores, data about objectives and their status, data about interactions with the learner, comments, very reduced learner information, suspend data and location, as well as entry and exit status.

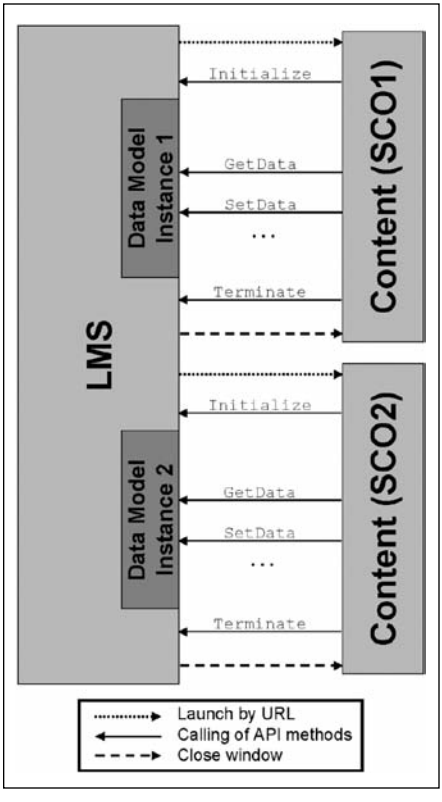

Figure 1. SCORM process diagram

The exchange of information issues are solved as follows (Figure 1).

Authentication. The LMS should provide an independent copy of the Data Model to each SCO, so authentication is transparent to them, since it is the LMS's mission to offer the correct user information when the SCO asks for it using the aforementioned methods.

Getting information about the user. A small amount of information about the user is stored in the Data Model, and intelligent content can access it using the method GetValue of the SCORM RTE API.

Storing the trace of user work. It can be stored in the interactions field of the Data Model. This element is useful when the interaction with the content can be stored in text mode and does not exceed the limit imposed by the standard. However, this can't be assumed, when dealing with intelligent content.

Registering end of work. When the user ends his/her work, the intelligent content can notify the LMS by calling the Terminate() method of the SCORM RTE API.

Sharing the results of the user's work. Undefined.

IMS Shareable State Persistence

To partially solve some of the constraints of SCORM RTE concerning intelligent content, the IMS Global Learning Consortium has created an extension to e-learning run-time systems, the IMS Shareable State Persistence (SSP) specification[8]. It enables the storage of and shared access to state information between content objects, which is needed in rich interactive intelligent content, such as simulations. It also defines how to apply this specification to SCORM-conformant LMSs.

Each learning object must declare the storage space that it needs – each contiguous zone of memory is called a *bucket* – as well as the persistence of the data, that is, whether the data is only available for the session, course or for all the courses taken by one learner. When the LMS selects a piece of content to be launched, it has to ensure that this object will have enough storage space when needed. The specification defines an API that allows the learning object to access the buckets it has declared as well as those that others have made visible for it.

As this standard is an extension to existent RTE standards, the only issues that it deals with concerning information exchange are the next ones:

Storing the trace of user work. It can be stored in buckets, using the space and the format needed by the intelligent content.

Sharing the results of the user's work. The information stored in the buckets can be shared by different pieces of intelligent content if the content that created the buckets defines them as shareable.

NONSTANDARDIZED SOLUTIONS

Taking into account the exposed constraints of the standardized solutions to deal with intelligent content, several research efforts have arisen to partially solve these problems. In this section, we study some of those nonstandardized systems, whose aim is to offer a distributed environment to deliv-

er adaptive education in different media. ADAPT^2 and MEDEA were developed for web-based education, whereas T-MAESTRO is an approach that has been designed to provide learning experiences in an Interactive Digital TV environment.

ADAPT^2

ADAPT^2[9] is an extension of the KnowledgeTree architecture (Brusilovsky & Nijha-van, 2002; Brusilovsky, 2004), a distributed architecture for adaptive e-learning based on reusable intelligent activities. KnowledgeTree has been used since 2002 to support instruction in several courses at the University of Pittsburgh.

Five main components comprise the ADAPT^2 framework (Figure 2).

1. The *Learning Portal* (LP) is similar to modern LMSs except that the content is not embedded in it. It organizes the learning material and provides students and teachers with the facilities necessary to participate in the learning process.
2. The *User Model Server* (UMS) is a centralized point in which are stored the user model of each user and the activities he/she has accomplished. This is taken into account by inference agents, which deduce the user's level of knowledge and other learning characteristics.
3. The *Ontology Server* – a major modification added to the former KnowledgeTree architecture – stores the ontological structures of the domain models and provides a centralized point to access con-

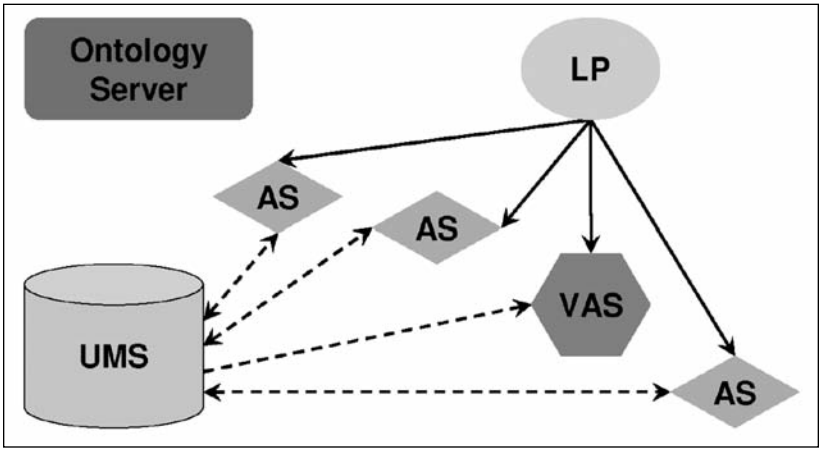

Figure 2. ADAPT^2 architecture

tent level metadata for all educational content, indexed with a specific Domain Ontology, that is the network of domain model concepts (topics, knowledge elements), which define the elements and semantic relationships between them. In addition, the Ontology Server provides a platform for exchanging high-level information about the students' knowledge, as calculated by different user model servers (from different applications which might have their local user modeling servers).

4. An *Application Server* or *Activity Server* (AS) implements one or more kinds of learning activities, which might be adaptive or non-adaptive. It typically implements a piece of intelligent content that engages the user in different series of activities over a period of time.
5. A *Value-Added Service* (VAS) adds some additional layer of services to the raw content provided by the Application Servers, for example, adaptive navigation support. The ADAPT^2 framework allows multiple instances of any type of its components, including user model servers. All of them are interchangeable, where the only requirement for this replacement is the common interface agreement.

Here is how major information exchange between components is organized.

Authentication. The portal provides a centralized single-login point for enrolled students to work with all learning tools and content fragments provided in the context of their courses (Figure 3). The portal uses simple authentication protocol to call the Activity Server, invoking a specific learning activity from the server using a unique URL. During the invocation process, the user parameters are passed to the intelligent server in the URL, using the HTTP GET protocol.

Getting information about the user. If the learning activity invoked by the portal needs information about the user, it requests this information from the UMS. This communication takes also place using the HTTP GET protocol: the Activity Server sends a specific URL to the UMS that includes user name, group, session ID and requested information. In response, the UMS returns the requested information in eXtensible Markup Language (XML) format.

Storing the trace of user work. Every semantically meaningful step of user work with a learning activity is reported to the UMS using another structured HTTP GET request. This step (event) is stored as part of user event history and is immediately used by inference engines to update the user model. UMSs can store a large number of events with rich details about each event.

Registering end of work. Not provided in published version.

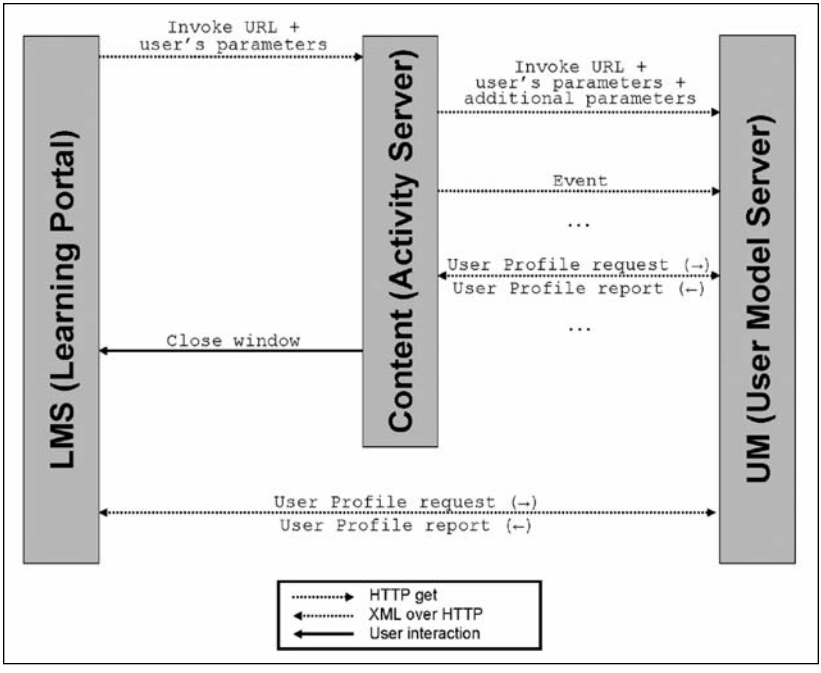

Figure 3. Communication flow for ADAPT[2]

Sharing the results of the user's work. All results of the user's work are stored in the user model on UMS in a standard format. Any other component of the framework can request the current state of the user model from the UMS.

MEDEA

MEDEA (Trella, Carmona, & Conejo, 2005) is an open learning platform for the deployment of intelligent and adaptive web-based educational systems. It is not a system that delivers content developed locally; on the contrary, it is a service-based learning platform that allows using and integrating different learning systems for educational and instructional purposes.

MEDEA's architecture (Figure 4) is composed of three main modules: the kernel, a set of Instructional Resources and the Connection Manager.

The *Kernel* consists of four different components.

1. The *Environment* is the student's interface, where a pedagogical task is executed, an instructional plan is requested and the student model can be consulted and modified.
2. MEDEA's *Student model* contains information about what the stu-

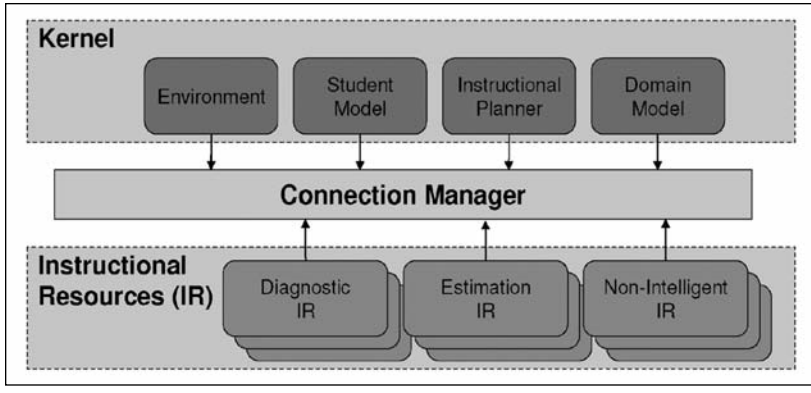

Figure 4. Architecture of MEDEA

dent knows about the subject and some other relevant characteristics. It includes the *student model updating service* that updates the model every time the student performs a pedagogical task.

3. The *Domain model* is the component that stores knowledge about the subject. It is defined by a semantic network of concepts, the relations between them and the pedagogical knowledge required for the learning process.

Guidance during the learning process is provided by

4. The *Instructional Planner*, which selects the concept to be studied as well as the most appropriate Instructional Resources (IR) to teach it. MEDEA defines three different types of IR: *Diagnostic Intelligent IRs*, which use some diagnosis process to establish the student knowledge level; *Estimation Intelligent IRs*, which use heuristics to estimate the student knowledge level; and *Nonintelligent IRs*, which do not have a student model. Each IR is implemented as a service that has a publicly accessible Web Service Definition Language (WSDL[10]) description that the connection manager will be able to use to communicate with the service. The Web Service Invocation Framework ([WIFS]; Duftler, Mukhi, Slominski, & Weerawarana, 2001), should be used to invoke those services dynamically at run-time.

For an intelligent IR to be available to MEDEA's users, three main features are required: (a) an overlay student model, (b) a set of semantic concepts in the domain model, and (c) a set of services implemented as web services and described in WSDL files – to initialize, end, register a new user, and export the user model.

Finally, the *Connection Manager* handles all the communications between

MEDEA's services. These communications can be viewed in two different levels: a software implementation level (the developer's level), reached by using emerging web services technologies[11] and a concept semantic level (the teacher's level). Concerning the latter, MEDEA requires that the platform shows a unique view to the student, where the integration of those different IRs is transparent to him/her. In addition, there should be a unified student model which collects and summarizes all the data from the different IRs student models. Therefore, the semantic relationships between the concepts of MEDEA and those of the IRs must be established. MEDEA provides an interface to perform this task manually, which is delegated to the teacher.

Next, we describe the mechanisms used by MEDEA to address the communication issues mentioned in the introduction section of this article.

Authentication. When a user begins to study an IR, MEDEA calls the User Registration Service and stores the userID and password assigned to this user by the IR (Figure 5). These parameters are passed as arguments to the IR every time the init service is called. This way, authentication is transparent to the user.

Getting information about the user. To obtain the student profile, the IR can access a service in the Student Model Manager that returns a list of concept/value pairs containing knowledge about the student, given a userID (MEDEA is in charge of translating this userID to the username introduced by the student when he/she entered the system).

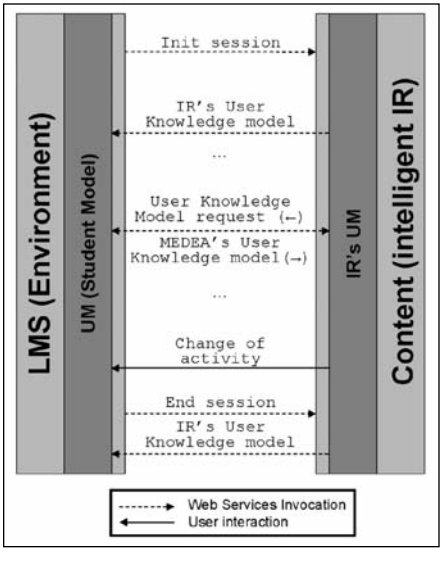

Figure 5. Communication flow for MEDEA

Storing the trace of user work. It is up to the IRs to keep the trace of user work, MEDEA does not provide any mechanism to store it and share it with other IRs.

Registering end of work. When a user selects a different task in the tree structure of the course, the end of his/her present work is registered by the system.

Sharing the results of the user work. Once the end of work is registered, the system calls the end service of the IR, which ter-

minates the task and calls the user model updating service to send a list of concept/value pairs for this user to MEDEA's user model, where all the information about the student knowledge is kept. This is the required way to update the user model; however, an IR can update the student model during its execution, calling the aforementioned service whenever it considers this necessary.

T-MAESTRO

T-MAESTRO (T-learning Multimedia Adaptive Educational SysTem based on Reassembling TV-Objects) is a system whose objective is to provide personalized learning experiences in the field of t-learning (Rey-López, Fernández-Vilas & Díaz-Redondo, 2006), that is, TV-based interactive learning[12]. Given that t-learning is still in an early state, the research efforts in this field should take into account the experience accumulated in e-learning to avoid making the same mistakes, such as e-learning's initial lack of standardization. For this reason, the authors of T-MAESTRO use a modified version of the SCORM standard in the design of their system (Rey-López, Fernández-Vilas, Díaz-Redondo, Pazos-Arias & Bermejo-Muñoz, 2006a, 2006b). T-MAESTRO is part of a broader project whose aim is to create a multichannel provisioning model for the residential environment, where multiple services will be accessed through a residential gateway. Today, the Open Service Gateway Initiative (OSGi[13]) Service Platform specification is the most adopted solution to solve the technological problem of building residential gateways and the platform over which T-MAESTRO is being developed.

The OSGi platform is a Service-Oriented Architecture that mainly consists of a Java Virtual Machine, a set of running components called *bundles*, and an OSGi framework. Bundles are the minimal deliverable application in OSGi. They are Java Archives that have a restricted lifecycle, which is externally managed by the OSGi framework. Aside from that, the minimal unit of functionality is really what OSGi calls a *service*. Thus, a bundle is designed as a set of cooperating services, which can be discovered after they are published in the OSGi Service Registry.

T-MAESTRO is then a set of *bundles* – each of them offers several services – that carry out together the task of offering personalized learning experiences to the student.

The fundamental building blocks of these personalized experiences are the self-adaptive SCOs (in accordance with the SCORM terminology): intelligent pieces of content that show different behaviors according to the student's characteristics, following a set of adaptation rules (Rey-López, Fernández-Vilas, Díaz-Redando, Pazos-Arias, et al., 2006a).

Figure 6 shows the architecture of the system, where two main components can be identified: The Learning Management System (LMS) and the API. The LMS is a set of four different bundles: (a) the selector, which offers

a service that decides whether a course is appropriate for the student or not; (b) the adapter, whose task is to modify the structure of the course according to the student's characteristics; (c) the Run-Time System (RTS), in charge of showing the content to the student, offering both launching and navigation services; and (d) the manager, whose mission is to intercommunicate between the other three services.

The API is in turn a bundle that implements the methods of the SCORM RTE API as services. Since T-MAESTRO does not work in a web-based environment, the API is provided as an OSGi bundle.

The API allows the different bundles of the LMS and self-adaptive SCOs to access an extended version of the SCORM Data Model, which includes several extra characteristics of the student called *adaptation parameters* (Rey-López, Fernández-Vilas, Díaz-Redondo, Pazos-Arias, et al., 2006a).

Figure 7 shows the communication process between the Learning Management System, the intelligent content and the User Model, which is part of the Extended Data Model (consisting of the adaptation parameters and a small amount of information about the user provided by SCORM) and is accessible through the API. After identifying the next SCO to be offered to the student, the LMS launches it by means of the method start. Once it is launched, it looks for the API in the OSGi Registry and notifies its state by means of the Initialize method. The SCO then starts to read the values of the adaptation parameters that it uses to self-configure – by means of the GetValue method – and sets the

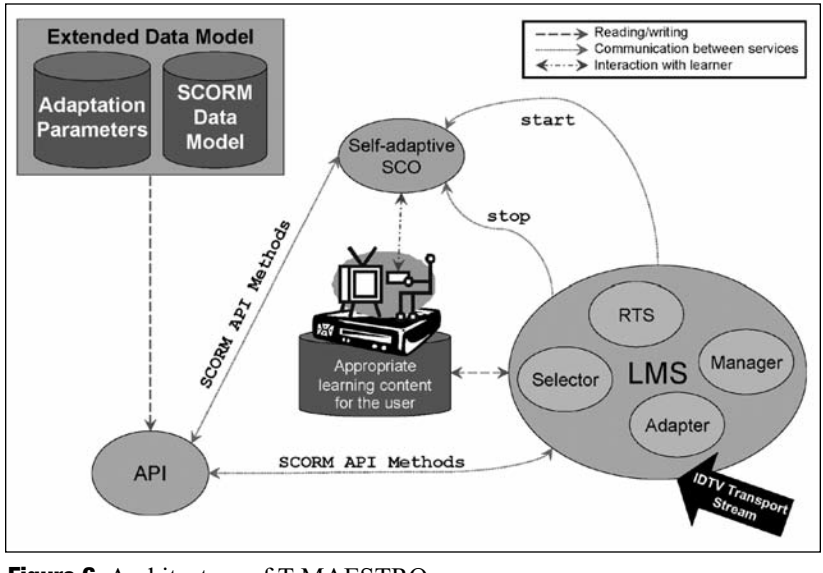

Figure 6. Architecture of T-MAESTRO

values of the correspondent SCORM Data Model elements according to interaction with the student – using the SetValue method. When it finishes its task, it informs the API with the Terminate method, which in turn alerts the LMS so that it can call the stop method of the SCO to close the latter.

The following paragraphs explain the mechanisms used by T-MAESTRO to solve the communication issues:

Authentication. Although T-MAESTRO inherits the SCORM architecture, the authentication cannot be transparent for the SCO since access to the user information stored in the Data Model takes place independently from the LMS. Due to the self-adapting nature of T-MAESTRO's SCOs, the LMS thus passes the userID to the SCO as an argument of the start method to allow it to ask the API for the characteristics of the appropriate user.

Getting information about the user. It is also slightly different from the standard, since it takes place by means of the API bundle, which contains the SCORM RTE API methods that have now a new argument: the userID.

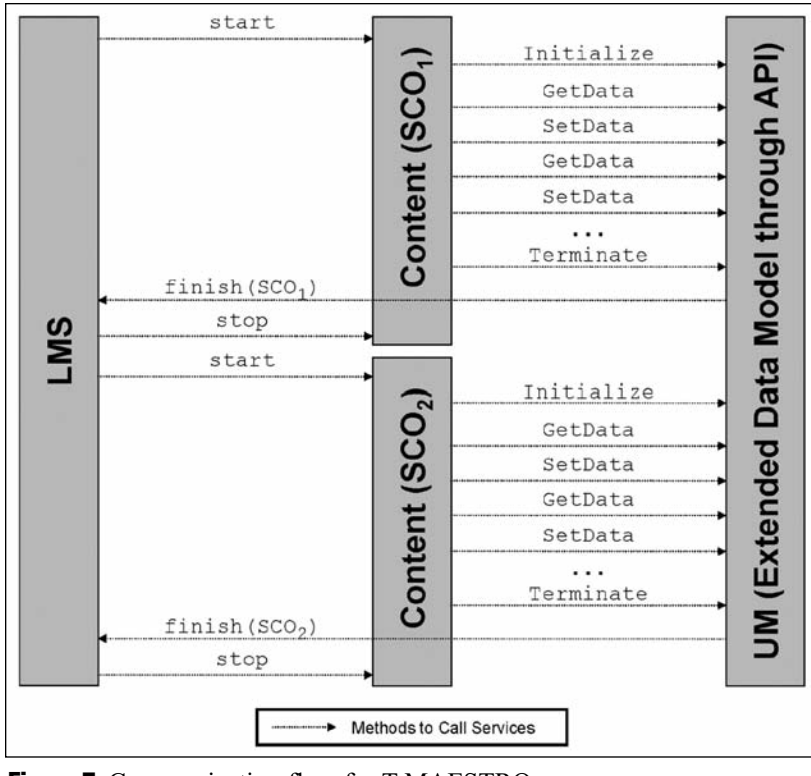

Figure 7. Communication flow for T-MAESTRO

Storing the trace of user work. As in SCORM, user work is stored using the interactions field of the Data Model. Hence, it suffers from the same constraints as SCORM.

Registering end of work. Once the user has finished working with the SCO, it notifies this state to the API using the Terminate method. The API then notifies the LMS so that it can call the stop method of the SCO and close the latter.

Sharing the results of the user work. An inference engine is added to the system to update the student data in the Data Model about the information written by former SCOs.

DISCUSSION, CONCLUSIONS AND FUTURE WORK

In this article, we have reviewed the needs of the learning systems that use intelligent content, presenting some standardized and nonstandardized solutions that intend to fulfill these requirements. These solutions have different ways of resolving the communication problems that are found when dealing with these types of learning objects (Table 1): user authentication, accessing and updating the user model by the intelligent content, storing the trace of the student's work and registering when they stop working on a task.

The standardized solutions studied have the great advantage that they permit content designers and LMS developers to work independently, since they provide reusability and interoperability. However, they do not currently offer an appropriate standardized way for the content to store the user's work trace or even the user profile information. Besides this, they do not provide any mechanism for updating it when the student interacts with intelligent content.

On the other hand, one of the standardized solutions' advantages is that they do not require any additional systems to launch the content except for a web browser and an LMS. Conversely, this fact makes the launch process very static, since the LMS cannot change the parameters the objects are invoked with to make them show different behaviors and the API should be provided in a predefined location, restricting the ability of the objects to open new windows.

On the flip side, the nonstandardized solutions overcome some of the former constraints, since they define their own systems to carry out the task of storing, managing, and updating information about the user. They even implement the different subsystems of the learning environment as independent pieces that can be easily replaced and reused.

Nevertheless, the reusability and interoperability of contents created to work in these systems are hard to achieve. In an effort to overcome these constraints, we are trying to integrate pieces of intelligent content initially devel-

Table 1
Comparing the Solutions

	SCORM RTE	IMS SSP	ADAPT²	MEDEA	T-MAESTRO
Authentication	Transparent	–	The user parameters are passed to the AS in the URL when invoked	MEDEA stores and manages the userID and password for each IR	The userID is passed to the learning object in its start method
Getting information about the user	Using the SCORM RTE API to access the appropriate fields of the Data Model	–	The UMS is asked about the data for a concrete userID invoking it by URL	The IR can access a service in the Student Model Manager to obtain a list of con-cept/value pairs	Using the SCORM RTE API (adding the userID parameter) to access the ex-tended Data Model
Storing the trace of user work	In the interaction field of the Data Model	In the buckets defined by the learning object	The content sends events to the UMS where they are stored	Not provided	In the interaction field of the Data Model
Registering end of work	Calling the terminate method of the SCORM RTE API	–	Not provided	Detected when the user selects a different activity in the course structure	Calling the terminate method of the SCORM RTE API
Sharing information about the user	Undefined	The information in the buckets can be shared by different learning objects	All the information about the user is stored in the centralized UMS	The IR is obliged to update the UM when it finishes (optionally updated during its execution)	An inference engine updates the student model using the information stored by learning objects

oped for different kinds of systems. So far, WADEIn, an Activity Server created in the context of ADAPT^2, has been successfully used as an Intelligent Resource in MEDEA. The next step in this process is to establish a set of common protocols for the intelligent content to communicate with LMSs so that these objects can be reused in all the systems that implement these protocols.

References

Aviation Industry CBT Committee (AICC). (2004). *CMI guidelines for interoperability, Revision 4.0*. Retrieved April 21, 2008, from http://www.aicc.org/docs/tech/cmi001v4.pdf

Brusilovsky, P. (2004). KnowledgeTree: A distributed architecture for adaptive e-learning. *Proceedings of the Thirteenth International World Wide Web Conference, WWW 2004* (Alternate track papers and posters, pp. 104-113), New York. New York: ACM Press.

Brusilovsky, P., & Nijhavan, H. (2002, October). *A framework for adaptive e-learning based on distributed re-usable learning activities*. Paper presented at the World Conference on E-Learning (E-Learn 2002), Montreal, QC, Canada.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996, June). ELM-ART: An intelligent tutoring system on world wide web. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *Proceedings of the Third International Conference on Intelligent Tutoring Systems, ITS-96* (pp. 261-269), Berlin, Germany. Retrieved April 21, 2008, from http://www.contrib.andrew.cmu.edu/~plb/ITS96.html

Butler, J. E., & Brockman, J. B. (2001). A web-based learning tool that simulates a simple computer architecture. *SIGCSE Bulletin – Inroads, 33*(2), 47-50.

Duftler, M. J., Mukhi, N. K., Slominski, A., & Weerawarana, S. (2001, October). *Web services invocation framework (WSIF)*. Paper presented at the OOPSLA Workshop on Object Oriented Web Services, Tampa Bay, FL.

IEEE Learning Technology Standards Committee (LTSC). (2004). *Application programming interface for content to runtime services communication.* IEEE Standard 1484.11.2. Retrieved April 21, 2008, from http://www.ieeeltsc.org/standards/1484-11-2-2003

IEEE Learning Technology Standards Committee (LTSC). (2005). *Data model for content to learning management system communication*. IEEE Standard 1484.11.1. Retrieved April 21, 2008, from http://www.ieeeltsc.org/standards/1484-11-1-2004

McKenna, A., & Agogino, A. (1997, November). Engineering for middle schools: A web-based module for learning and designing with simple machines. *Proceedings of FIE'97, Frontiers in Education Conference* (pp.1496-1501), Pittsburgh, PA. Champaign, IL: Stipes.

Rey-López, M., Fernández-Vilas, A., & Díaz-Redondo, R. P. (2006, June). *A model for personalized learning through IDTV*. Paper presented at the Adaptive Hypermedia, Adaptive Web-Based Systems 2006 (AH2006), Dublin, Ireland.

Rey-López, M., Fernández-Vilas, A., Díaz-Redondo, R. P., Pazos-Arias, J. J., & Bermejo-Muñoz, J. (2006a, July). *Adaptive learning objects for t-learning*. Paper presented at the the 5th International Conference on Web-Based Learning (ICWL 2006), Penang, Malaysia.

Rey-López, M., Fernández-Vilas, A., Díaz-Redondo, R. P., Pazos-Arias, J. J., & Bermejo-Muñoz, J. (2006b, October). *Extending SCORM to create adaptive courses*. Paper presented at the First European Conference on Technology Enhanced Learning (EC-TEL 2006), Crete, Greece.

Trella, M., Carmona, C., & Conejo, R. (2005, July). MEDEA: An open service-based learning platform for developing intelligent educational systems for the web. *Proceedings of the Workshop on Adaptive Systems for Web-Based Education at the 12th International Conference on Artificial Intelligence in Education* (AIED'05; pp. 27-34), Amsterdam, The Netherlands. Amsterdam, The Netherlands: IOS Press.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence and Education, 15*(3), 147-204.

Acknowledgement

Partly supported by the R+D project TSI 2007-65599 (Spanish Ministry of Education and Science), the KESP project funded by FÁS of Ireland, and the National Science Foundation of Ireland through E.T.S. Walton Visitors Award.

Notes

[1.] http://ieeeltsc.org

[2.] http://www.imsproject.org

[3.] http://www.aicc.org

[4.] http://www.adlnet.gov

[5.] http://www.cen-ltso.net

[6.] http://www.adlnet.gov/scorm

[7.] The SCORM standard defines a Sharable Content Object as a learning object that has the ability to communicate with the LMS.

[8.] http://www.imsglobal.org/ssp

[9.] http://www.sis.pitt.edu/~paws/project_adapt2.htm

[10.] http://www.w3.org/TR/wsdl

[11.] http://www.w3.org/2002/ws

[12.] http://www.pjb.co.uk/t-learning

[13.] http://www.osgi.org