Towards the Reuse of Practical and Collaborative Learning Experiences

Manuel Caeiro-Rodríguez, Jorge Fontenla-González, Martín Llamas-Nistal, Luis Anido-Rifón
University of Vigo, School of Telecommunications Engineering
Vigo, E-36310 {Manuel.Caeiro; jfontenla; Martin.Llamas; Luis.Anido}@det.uvigo.es

Abstract – In this paper we introduce a proposal to support the reuse of educational resources involved in practical and collaborative learning experiences, intended to complement existing reuse initiatives for LMS resources. These initiatives are mainly focused in media contents, while our proposal is devoted to the tools that can be used in practical educational scenarios (e.g., simulators, remote labs, communication tools) and, more important, to the ways such tools are intended to be used. A main point of the proposal is to enable the use of different tools. In relation to the intended use, three main issues are described: authorizations, awareness and interaction. The proposal is part of a more comprehensible development based in the language PoEML. This language extends the IMS Learning Design capabilities to support practical scenarios through a generalized solution and following an approach based on the separation of concerns.

Index Terms — Educational Modeling Languages, IMS Learning Design, Reusability, Interoperability, Practice Learning, Collaborative Learning.

INTRODUCTION

The sharing and reuse of educational resources has been a main issue in the e-learning community. As a result, the Learning Object (LO) [1] concept and related standards (e.g., SCORM [2], LOM [3], QTI [4]) have been developed. Basically, these proposals define how to structure and arrange digital media contents to support and facilitate the development of learning experiences in Learning Management Systems (LMSs). Nevertheless, collaborative and practical learning experiences involve more than simple media contents. These learning experiences cannot be developed by simply considering the digital media, but they need to arrange roles in accordance with a certain structure (e.g., a moderator, a speaker and an audience), to provide environments with appropriate applications and services (e.g., a bulletin board, a simulator, an e-mail service), to control the actions that can be performed by each role (e.g., to write), to control the interactions that can be produced in a discussion (e.g., to initiate a new chat session), etc. All this information should be arranged in accordance with well-defined models to enable their computational support and eventual reuse.

This paper introduces a proposal to support the computational description and reuse of these issues. It is part of a more comprehensible language conceived to extend the capabilities of current Educational Modeling languages (EMLs). EMLs [5] [6] have been proposed as languages that support the modeling of educational practices (e.g. a course, a lab practice, a collaborative seminar). In 2003, the IMS Global Consortium published the Learning Design (IMS-LD) Specification [7] [8], currently considered as the de facto standard EML. The main idea underlying EMLs is to provide an only language rather than several languages to describe educational units in accordance with different pedagogical approaches. If several languages are considered, each one requires specialized implementations of both design and runtime systems. Using a single EML, for example IMS-LD, a single set of design and runtime tools would be enough to support the desired wide range of pedagogies. If the development of IMS-LD educational materials is easy, then the reusability required to solve the scale-production of educational materials for different pedagogies is also solved.

As a result, EMLs offer a promising view for the future development of e-learning systems. Nevertheless, the description of a large number of the issues involved in collaborative and practical educational units is not enabled in existing EMLs. They provide support to describe common scenarios, but not to describe specific situations.

The proposal introduced in this paper is devoted to support the computational description and reuse of practical and collaborative learning experiences in a broad sense. This proposal is in some aspects quite similar to the IMS-LD ones, but they provide a more generalized solution. Particularly, three specific parts, named as perspectives, are introduced: the authorization perspective, about the assignment of different permissions to different participants; the awareness perspective, about the capture of events, their processing (e.g., filtering, aggregation, correlation) in order to capture relevant information and the notification to interested participants; and the interaction perspective, about the invocation of operations in applications and services in an automatic or semi-automatic way, in accordance with the satisfaction of certain conditions or the trigger of some event.

Next section introduces an example that describes a typical problem demanding the reuse of practical educational scenarios. Then, the main requirements involved in this challenge are analyzed. The solution proposed is described in the next section and next the eventual business model derived from it. The paper finishes with a comparison of these proposals with related works and some conclusions.
This section introduces an example to describe the main problems approached in this paper. The example is about a Web-based educational system used to support the practical part of a “Computer Architecture” subject, in the Engineering Telecommunication School at the University of Vigo. This system, named as SimulNet [9], was produced some years ago to facilitate the development of educational practices using simulators and communication tools. These practices involved the performance of practical learning activities and some collaboration involving learners and tutors.

SimulNet provides learners the chance to perform their practical assignments outside the traditional laboratory room. In addition, the SimulNet system also provides several functionalities in order to control and manage educational activities. Nevertheless, learners are provided with a good degree of tutor support and supervision. There are three main types of educational scenarios supported by this application:

- **Free individual scenarios.** In some cases, the learner is intended to interact freely with a simulator. This is a common use when the learner is asked to solve some problem. The problem is provided as a simple text file, containing a description of what the learner has to do. In addition, from an academic point of view, despite the learner is intended to work alone, it is important that tutors receive information about what learners are doing and what they have done in their interaction with the simulator, namely: learners’ awareness. Therefore, an event-management system was developed to support the capture of important actions performed by the learners and results produced in the simulators. These events were processed (e.g., filtered, aggregated) in order to provide tutors with information as accurate as possible, to facilitate the detection of possible problems and the assessment of learners’ activities.

- **Collaborative scenarios.** In some cases, a simulator is intended to be used by several learners and/or tutors at the same time. For example, a learner may be required to solve some problem under the supervision of a tutor, as in an examination. These scenarios usually include some synchronous communication tools, such as a chat or a whiteboard, in order to facilitate the interaction among participants. For example, the tutor could ask some questions to the learner. A main issue is that different users should be granted different permissions to act on the simulator and on the communication tools. For example, the tutor could be assigned permissions to moderate the conversations in the chat, while learners can write information only when the tutor decides they can do it.

- **Guided scenarios.** Some practical scenarios include guided simulations in order to demonstrate a certain practical use. For example, the first lab lecture is about how to create and run a simple application in the simulator. This involves the performance of a series of steps and actions that can be arranged as a script: initialize the memory and the registries, edit a program, compile, execute, etc. Scripts involve the performance of automatic or semi-automatic actions, but they may also require some interaction from the user [10]. For example, to pause and run the simulation, to introduce some parameters, etc.

These scenarios are supported by the SimulNet Web-based application in a proper way. Nevertheless, it presents reusability and scalability problems. Each year, we need to modify the practical assignments for the learners. Some issues involved in this modification are very simple, such as the problem description text files. Nevertheless, some times we would like to change the awareness received from the learners, or the permissions assignment, or to introduce new guided simulations. A more dramatic change is to introduce a new simulator, as it does not simply require a new software program, but also changes in the SimulNet system logic to control and manage the new simulator. This type of changes requires to re-program certain modules of the system. Unfortunately, this is a time and resource-consuming task.

Similar problems appear when the application has to be transferred to a different institution, as for example to the UNED University, the Spanish national distance education. In this case, the complete application has to be installed in the same way as it is in the University of Vigo. In addition, teachers have to adapt their practices to the programmed scenarios. Anyway, this is not a real solution, as teachers (mainly if they belong to institutions so different as a face-to-face engineering school and a distance university) need to introduce specific scenarios in accordance with their preferences and their learning requirements.

**MAIN REQUIREMENTS**

In previous works we have analyzed the needs involved in practical learning, mainly in engineering courses, in order to identify a clear set of requirements [11, 12, 13]. The introduced example describes a typical scenario gathering many of the identified requirements. They range from organizational issues, to the support of collaborative and interactive environments and practices. The following list shows the main requirements identified to support practice based scenarios:

- **It should be possible to deploy environments including certain tools and data elements.** It should be possible to include different kinds of tools: (i) autonomous applications that do not require installation and applications or (ii) services that need to be installed and configured by hand prior to their use. Therefore, these second type of applications can not be transferred together with other course contents in a learning package [6].

- **It should be possible the assignment of different permissions to different participants.** The tools and data elements included in an environment may require several permissions. For example, data elements usually involve read and write permissions.

- **It should be possible to capture and to process events in order to obtain relevant information about the environment operation.** The tools and data elements included in an environment can generate several events in
response to the users’ interaction. For example, a synchronous communication tool can issue events such as “participant added” and “new message”. In addition, it should be possible to include parameters in the events. These parameters should enable some processing functionality (e.g., filtering) to facilitate the gathering of relevant information.

- It should be possible to invoke operations in accordance with predefined scripts. The tools and data elements included in an environment can offer certain operations. For example, a simulator can offer operations “to load a simulation file” and “to execute the simulation”. The script could require a certain level of interactivity with the user, invoking operations in response to certain user-related events. For example, a simulation may be loaded and executed when the learner demands to see a guided simulation. In addition, it should be possible to include parameters in the operations (input and output). These parameters should enable to transfer data from the environment data elements to the tools.

These are the main requirements of the problem discussed in this paper. Nevertheless, they are a part of a more comprehensible development named PoEML, that also involves theoretical and collaborative learning scenarios. PoEML (Perspective-oriented Educational Modeling Language) [11] is a modeling language for educational units supporting the description of different learning approaches, similarly to IMS-LD. Nevertheless, from the point of view of the proposal presented in this paper, PoEML allows the description of generic practical scenarios, allowing the use of different applications and the description of their operation, namely: the management of permissions, events and operations.

**MAIN COMPONENTS OF A SOLUTION TO SUPPORT PRACTICAL EDUCATIONAL SCENARIOS**

PoEML approaches the support of educational scenarios by following a very structured approach. In the case of practical scenarios, PoEML enables to indicate the tools and data involved, the required participants, and separate specifications about the authorizations that need to be assigned, the events that need to be captured and their processing, and the operations that need to be invoked. Figure 1 shows part of the XML code describing the subject introduced in section 2.

The different parts of the course are represented as Educational Scenarios (ESs). In this example, there is a main ES (ESRoot), representing the whole course, that is made up by a “Theoretical” ES and a “Practical” ES. In addition, the “Practical” ES is made up by other two ESs: an “8 bit Simulator Practice” and a “32 bit simulator practice”. Each one of the practical ESs involves particular Tools such as simulators and synchronous communication tools, AuthorizationSpecifications indicating how the permissions have to be assigned to the actors involved, AwarenessSpecifications indicating what events need to be captured and how they need to be processed, and InteractionSpecifications indicating what operations need to be invoked and when.

**FIGURE 1**

XML code fragment from the Computer Architecture subject.

AuthorizationSpecifications simply enable to indicate the permissions that have to be assigned to each participant. This assignment involves: firstly, selecting the source of the permission, that can be a data element, a tool or a service, or a specific instance of a certain tool or service; secondly, indicating the permission assigned; and thirdly, indicating the role or role instance (e.g., all the learners or a specific learner) granted with the permission.

AwarenessSpecifications enable to indicate how events have to be captured, processed and notified. Similar to the AuthorizationSpecifications they also involve: firstly, to select the source of the event, that can be a data element, tool or service, or a specific instance; secondly, to indicate the event to be captured; thirdly, to indicate how such event or set of events should be processed; and finally, to indicate the person or control (e.g., a Event-Condition-Action) that should be
particularly the figure represents 6 main components of the modular approach. Figure 2 shows the main building blocks of run-time system supporting their execution also follows a Specifications namely, it is not necessary to indicate a reference (e.g., URI) double purpose. Firstly, to enable the late binding of tools, these descriptions are included in the course XML with a example, if the first practice wants to be copied, but a different specifications can be provided later. Secondly, to know during the design-time what permissions, events and operations that are used to indicate how each specification is going to be activated. During execution, a specification is applied and used by the system if it is activated. In this way, the more complex part of these specifications is the processing of the events. Events can be generated in a dynamic and unexpected way. The processing of events can involve filters (e.g., in order to capture events that satisfy a certain condition), aggregations (e.g., in order to compute the number of events detected), and correlations (e.g., to detect if two events are produced at the same time). Therefore, the edition of these specifications is not simple. Anyway, the description of the capture and processing of events in these terms is much more flexible from the LMS point of view.

InteractionSpecifications enable to indicate how operations have to be performed in applications and services, what parameters have to be used and when they have to be invoked. Similar to the previous cases they also involve: firstly, to select the “source” of the operation, that can be a data element, tool or service, or a specific instance; secondly, to indicate the operation to be invoked; thirdly, to indicate the parameters that should be used in the operation; fourthly, to indicate how a set of operations should be arranged (e.g., sequence, parallel, if-then-else, loop); and finally, to indicate the person or control (e.g., a Event-Condition-Action) that determines when the operation has to be invoked. Notice that the three kinds of specifications have attributes activated and participantActivation. These attributes are used to indicate how each specification is going to be activated. During execution, a specification is applied and used by the system if it is activated. In this way, the descriptions are very flexible, as different specifications (e.g., different permission assignments) can be applied changing the behavior of the system and therefore the educational practice. This feature also facilitates the reuse, as the several specifications can be changed without affecting to others. For example, if the first practice wants to be copied, but a different permission assignment is required we only need to change the AuthorizationSpecification element.

It is also important to highlight that the ESRoot includes a description of the tools involved in the practical Es. These descriptions involve the specification of Features, Permissions, Events and Operations, available in such tools. These descriptions are included in the course XML with a double purpose. Firstly, to enable the late binding of tools, namely, it is not necessary to indicate a reference (e.g., URI) to a tool during the design-time as a tool satisfying the descriptions can be provided later. Secondly, to know during the design-time what permissions, events and operations that can be used in the Authorization, Awareness and Interaction Specifications.

Similarly to the language used to describe the courses, the run-time system supporting their execution also follows a modular approach. Figure 2 shows the main building blocks of such an LMS related with the support of practical scenarios. Particularly, the figure represents 6 main components of the LMS:

- The Data component is responsible for the maintenance and management of the data elements (e.g. files) available in the Es. This component is accessed by the Awareness and Interaction components in order to interchange data from the parameters of events and operations, respectively. In this way, it is supported the data communication between the tools and the LMS.

- The Environment component supports the arrangement of the practical scenarios. Basically, it provides a support to maintain data elements, tools and the three types of specifications.

- The Tool component supports the provision of the tools including functionalities such as late-binding, session control and management.

- The Authorization component is responsible for the management and assignment of permissions to participants.

- The Awareness component is devoted to support the capture of events, their processing (e.g., filtering), and their notification to interested recipients.

- The Interaction component determines the invocations of operations in accordance with predefined time points and events.

The system includes a Tool Server that supports the search and execution of tools separately from the LMS. This solution enables the use of different tools from different locations. In order to support the interaction between the LMS and the Tool Server, the system requires two main elements. The first one is a Broker to support the search and access from the participants (links 1 and 2), including Authentication and Access Control functionalities. The second component is a

\[ \text{Figure 2: Architecture and Main Components for the Provision of Services to LMSs.} \]
Instruments with this model, user applications do not need to be installed computing is to provide an alternative to having local servers shared computing resources. The main idea underlying cloud ideas of grid computing and utility computing for the use of user interface, both for the learner and for the teachers, of the permissions.

**THE BUSINESS MODEL**

Nowadays, many organizations are developing numerous tools to be used in practical educational scenarios (e.g., communication tools, software labs, virtual Web labs, remote Web labs). These tools are programmed with different programming solutions (e.g. AJAX, J2EE, MATLAB, LABVIEW) and designed with their own authentication service, control service, user management service, etc. In this way, these tools do not take advantage of the functionalities already provided by LMSs [14]. The architecture introduced in the previous section enables the integration of this variety of tools in the LMS. Tools can be developed by different participants and using different technologies. They need to support the interfaces introduced to enable the communication and interaction with the LMS.

Figure 3 shows a representation of a typical software configuration in accordance with our proposal. Different tools could be developed and maintained in separate Tool Servers. In the figure, virtual tools (Server A) and remote tools (Server B) are represented. Learners and teachers would access these services through the LMS. In addition, it is possible that different LMSs would use the same Tool Server. Therefore, the provision of LMS and tool functionalities can be developed and maintained separately, while their integration is determined by the introduced proposal.

This model fits in the cloud computing approach [15] promoted by Google and IBM. This approach is based on the ideas of grid computing and utility computing for the use of shared computing resources. The main idea underlying cloud computing is to provide an alternative to having local servers or personal devices handling users' applications. In accordance with this model, user applications do not need to be installed and executed remotely in servers and accessed by users through a general purpose Web browser. This is possible as connectivity costs fall and as evolution of processors architectures enables the massive provision of computational capabilities. The provision of user applications by remote servers allows the sharing of computational resources and the separation of infrastructure maintenance duties from the domain-specific application development, in this case, LMSs and e-learning applications and services.

**RELATED WORK**

During the last years many EMLs have been developed. The most relevant one has been IMS-LD, focusing the main research efforts in this domain. However, other languages have been proposed previous to the apparition of IMS-LD: OUNL-EML [16] (developed in the Open University of the Netherlands, it was taken as the base language for the development of IMS-LD), MISA-MOT [17], E2ML [18]. Many other languages have been proposed after the publication of IMS-LD as approved IMS Specification as well: the CSCL Script formalization [19], Xedu [20], CPM [21], coUML [22], etc.

The languages proposed previous to the publication of IMS-LD provide minimum support for the integration of tools. Basically, they enable to include tools in the learners’ working environments, but they do not recognize the need of specifying how such tools should or must be used. The OUNL-EML enables the use of several object types, where some applications are identified: tool object, search object, communication object, announcement object, monitor object and test object. The most interesting idea is introduced in MISA-MOT, that suggests the definition of rules of collaboration and execution, but they have not been further detailed neither developed.

The IMS-LD specification enables the description of general activities and environments involving any kind of Web-based applications. Nevertheless, similar to the previous languages, it does not further consider the support of their use by learners and teachers. In addition, it also enables to perform the specification of four services: send-mail, index/search, monitor and conference. Related with these services there are some elements similar to the ones proposed in PoEML. For example, the send-mail includes an operation to send a notification, similar to the operations recognized in the interaction perspective. Another example: the conference service includes specific roles such as moderator, similar to the roles recognized in the authorization perspective. In addition, the monitor service is able to indicate the capture of some events. Therefore, most of the concepts introduced in our proposal are recognized in IMS-LD, but PoEML supports them in a more general and decoupled way. Related with IMS-LD it is also important to notice that there have been some efforts to provide an implementation to the integration of tools in the SLED project (http://sled.open.ac.uk).

The EMLs proposed after the publication of IMS-LD do not make important contributions to enhance the specification
of tools and their use. PoEML resumes the efforts to design and take to practice reusable educational experiences.

CONCLUSIONS

We believe that the development of scale-production e-learning solutions supporting different pedagogies is possible, but it requires the revision of current LMSs’ architectures. As EMLs propose, the models of educational units can be used to drive the operation of appropriate e-learning systems. In this way, while EMLs support the modeling of educational units in accordance with different pedagogical approaches, the corresponding e-learning systems can be also used to support education in accordance with different pedagogical approaches, enabling in this way to provide an appropriate support to different academic institutions in different contexts. These e-learning systems will process the models of educational units proposing to each participant with the goals he/she is intended to perform, including the information and applications required, and in accordance with a certain order, temporal restrictions, etc.

Our proposal is intended to obtain a simpler, more adaptable and flexible EML. Particularly, the solutions proposed facilitate the reuse of models of educational units, enabling the description of different alternatives and their activation to control the changes. In addition, those changes can also be performed dynamically during the execution. This is very important as many times educational plans have to be changed and updated during run-time. The proposal also outlines the main building blocks of a run-time engine based on such EML, inheriting from it its modularity and adaptability. However, we must point that the architecture of such an engine is still being taken to practice, due to the early stage of its development.

ACKNOWLEDGMENT

The authors would like to acknowledge for its partial support the Spanish Ministerio de Educación y Ciencia under grants TIN2007-68125-C02-02 and TSI2005-08225-C07-03, and the Galician Consellería de Innovación e Industria under grant PGIDIT06PXIB322270PR.

REFERENCES