A General Architecture for the Authoring and the Operationalization of e-Learning Applications with Educational Modelling Languages

Iván Martínez-Ortiz[1], Pablo Moreno-Ger[2], Jose Luis Sierra[2], Baltasar Fernández-Manjón[2]

[1] Centro de Estudios Superiores Felipe II, Aranjuez, Madrid, Spain
`imartinez@cesfelipesegundo.com`
[2] Department of Computer Systems and Programming, Universidad Complutense de Madrid, Fac. de Informática, 28040, Madrid, Spain
`{pablom, jlsierra, balta}@fdi.ucm.es`

Abstract. The modelling of the educational processes and their operational support is a key aspect in the construction of more effective e-learning applications. Instructional models are usually described by means of an *educational modelling language* (EML). The EML used can be one of the available standards (e.g. IMS Learning Design), the customization of a standard to meet a specific application profile, or even a domain-specific EML specifically designed to best fit in the very particular needs of a learning scenario. In this paper we propose <e-LD>, a general authoring and operationalization architecture capable of dealing with all these possibilities in a highly modular and flexible way. We also outline a specific implementation of <e-LD> based on standard XML technologies and in the BPEL4WS workflow management language, and we describe how this implementation can be used to support IMS Learning Design.

1 Introduction

The representation of the learning content in the form of potentially reusable *learning objects* [14], adapted and assembled in many different learning scenarios is a very important aspect of an e-learning application, but it is only a piece of the whole puzzle. Indeed, the application should also coherently integrate resources and participants such as students or instructors across a well-defined set of learning activities that are structured carefully and deliberately in a *learning workflow* to promote more effective learning. In other words, e-learning applications must be based on a well-founded *educational process*. To achieve this objective, these learning processes must be modelled with the level of detail required for the development of a computer artefact. For this purpose, a suitable *educational modelling language* (EML) can be used [15]. An EML is a domain-specific language specially suited for describing instructional processes. A good example of EML is IMS Learning Design (IMS LD) [10,8].

Depending on the degree of formalization, the EML can support an operational semantics, and therefore it can be interpreted by a *player* in order to automatically produce e-learning applications from the models and other resources (e.g. learning material) referred to in these models. Typical examples of players for IMS LD are Coppercore [4] and RELOAD Player [16]. On the other hand, these instructional models should be provided by educators. Although EMLs are domain-specific, and therefore they integrate concepts close to the domain of expertise of these educational experts, the objective of making these languages directly executable by computers hinders their usability. For instance, it is not reasonable to require an educator to directly represent his/her instructional designs in the XML encoding of IMS LD, even with the help of XML editing tools (e.g. XML Spy). For this purpose, authoring tools must be provided (e.g. RELOAD Editor for IMS LD).

In addition, instructional models on their own are not sufficient enough for generating full e-learning applications. Indeed, it is also necessary to provide operational support for the basic activities introduced by the model. While some of these activities can be implemented using the standard support provided by a browser, other more specialized activities can require domain-specific support. Moreover, while the use of a standard EML (e.g. IMS LD) makes the reuse of a player possible for each application, the EML itself can evolve to a new version, or its operational semantics, and even the language itself, can be specialized to meet the needs of particular application profiles. Finally, some learning domains can require more specialized EMLs (if there is a single lesson learned in Computer Science is that there is not *universal* solutions). Thus the authoring and the execution environments must be able to rapidly react to these changes. In this paper we propose <e-LD>, an authoring and execution architecture equipped with the required flexibility.

The rest of the paper is organized as follows. In section 2 we describe our <e-LD> architecture from a conceptual point of view. In section 3 we propose a concrete implementation of <e-LD> based on standard XML technologies and BPEL4WS, a workflow management and web service orchestration language. In section 4 we outline how this architecture is used to support a specific EML (i.e. IMS LD). Section 5 outlines the related work. Finally, in section 6 we give the conclusions and the lines of future work.

2 The <e-LD> Conceptual Architecture

The conceptual architecture for <e-LD> is outlined in
Fig. 1. This architecture, which is conceived to accommodate the different sources of variability identified in the introduction, distinguishes the following components:
-   The *units of learning*. The main goal of the architecture is to produce, e-learning applications based on units of leaning. A unit of learning is formed by a *learning design* that is a formal description of a particular educational process, and a set of *learning resources*, which are other information items required for the model to work. Notice that, although the terminology used is to some extend borrowed from IMS LD, here we are using it in a conceptual and a representation-agnostic sense.

- *Authoring tools*. These tools are used by learning designers to simplify the production of formal learning designs that are integrated in the units of learning.
- The *Execution Platform*. This platform integrates the *basic activities* required to execute a unit of learning, as well as a *workflow engine*, which is used to orchestrate the execution of these activities. As recognized in the IMS-LD specification [8] and in works like [13,22], learning designs can be viewed as particular cases of *workflows*, as defined in the context of business process integration [5]. Therefore, <e-LD> adopts a workflow management system as the basic execution environment for the learning designs. The system provides a suitable *workflow language*, which is used to describe workflows that will be interpreted by a *workflow engine*. The chosen workflow language will be to <e-LD> as an assembling language is to a compiler of a high-level programming language.
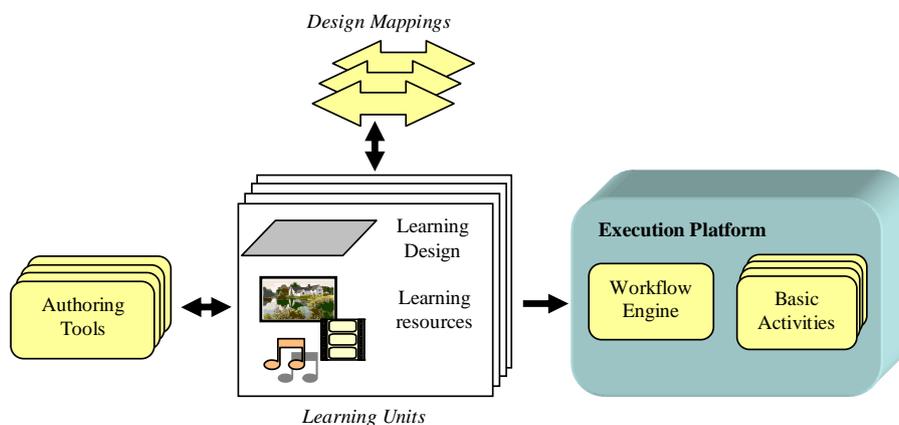

Fig. 1. The <e-LD> conceptual architecture

- *Design mappings*. These components are used to translate a learning design from a source representation to a target one. Design mappings are used to connect the other components together. This way, the design can be initially encoded in an authoring-specific format, then translated into a suitable EML using a design mapping, and the resulting representation can be translated into a workflow-oriented language using another mapping. In the same way, design mappings can be also used to connect authoring tools together. In this case, the EML is used as a common representation for the mappings involved. It is interesting to notice that the architecture does not require these mappings to be entirely automatic. Instead, they can comprise intervention of the user (e.g. in order to translate a complex or not entirely formalized EML in a suitable workflow representation). This possibility also promotes a rational separation of roles in the development process (e.g. collaboration between instructors and EML experts during authoring, and collaboration between learning designers and developers during operationalization).

3 Implementing the <e-LD> Architecture

We are implementing the <e-LD> architecture in the context of our XML based <e-Aula> experimental Learning Management System [20]. Besides, we have chosen BPEL4WS [2] as the workflow language. The resulting implementation is outlined in Fig. 2.

In this implementation learning designs must be represented using suitable XML-based domain-specific languages. Notice that this is not a severe constrain, since XML is currently being adopted as an standard interchange format between tools, and the occasional non XML-compliant authoring tool can be wrapped to produce an XML-based representation. Besides, it is also a common practice for the existing EMLs to provide an XML binding, and it also facilitates the provision of EMLs tailored on specialized domains following a document-oriented approach, such as those described in [18]. Finally, BPEL4WS itself has also a XML.

As a consequence of the use of XML, the design mappings are conceived as transformers between XML-based markup languages. Notice that the design mappings can be provided using standard XML processing technologies and many of these mappings can be given using transformation languages like XSLT. For more complex mappings, it could be possible to use standard XML processing frameworks (e.g. DOM or SAX), and even more sophisticated solutions for the incremental operationalization of XML-based domain-specific languages [19]. Finally, since basic activities in BPEL4WS are represented as web services, the implementation of the basic activities in the architecture must be provided by a web-service programmatic interface. Thus, the resulting applications will exhibit a service-oriented architecture.

4 Supporting IMS LD in <e-LD>

IMS LD has been readily integrated in <e-LD>. This section describes this integration, which has been performed reusing existing tools and technologies in order to decrease the overall cost as it is further explained in [11].

The authoring of IMS LD descriptions is largely done using UML activity diagrams (Fig. 3). In these diagrams IMS LD activities are represented by means of UML activities. Sequencing is expressed using transitions, synchronization bars and diamond shapes. Activity structures are represented using simple sequencing of the corresponding UML activities, when the activities in the structure must be executed in sequential way (as *Grade Exam* and *Evaluate* in Fig. 3), or by using a number of transitions when they must be chosen in random order (as *Prepare Test* and *Prepare Questionary* in Fig. 3). Finally, assignments of roles to activities are represented using swimlanes in the activity diagrams (as *Teacher* and *Student* in Fig. 3). Besides to these dynamic aspects, in our integration of IMS LD the static aspects (e.g. *environments*, *conditions*, and the definition of the prerequisites/objectives and learning contents to be presented in the atomic activities) are edited using RELOAD editor.
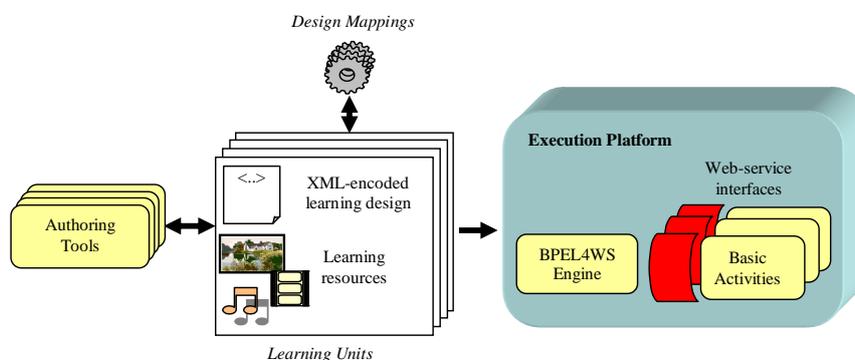
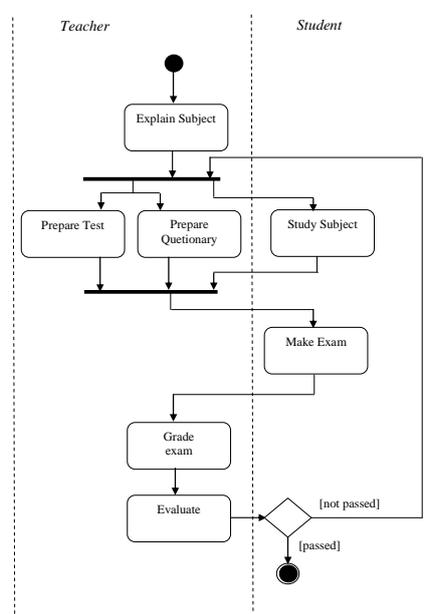Fig. 2. <e-LD> BPEL4WS-based implementation

Fig. 3 UML Activity Diagram representing a simple educational scenario

Regarding operationalization, the dynamic parts of IMS-LD (*method - activity* structures) have a correspondence with the flow control structures of BPEL4WS and the static parts of the learning design are used as configuration parameters for the web services to be orchestrated by the BPEL4WS process. Some of these web services will be reusable across many different learning scenarios, while others will be specific for a concrete application. In addition, many of these web services will be fed with the appropriate learning resources during the initialization stages.

The architectural details of the integration are depicted in Fig. 4:

- The authoring of the activity diagrams can be performed with any tool supporting XMI (an XML-based specification which allows interchange of UML models between tools as XML documents) as interchanging format. Besides, the represen-

tation conventions described below allows for the provision of a design mapping for generating IMS LD-aware representations from XMI documents containing activity diagrams, and also another mapping for performing the inverse step (i.e. for representing any IMS LD-aware learning design as an XMI document that can be edited with the UML editing tool). These two mappings are largely based on XSLT stylesheets, although tackling a few issues requires more conventional programming (e.g. dealing with the IMS Package Interchange Format packaging the learning of units in this implementation).
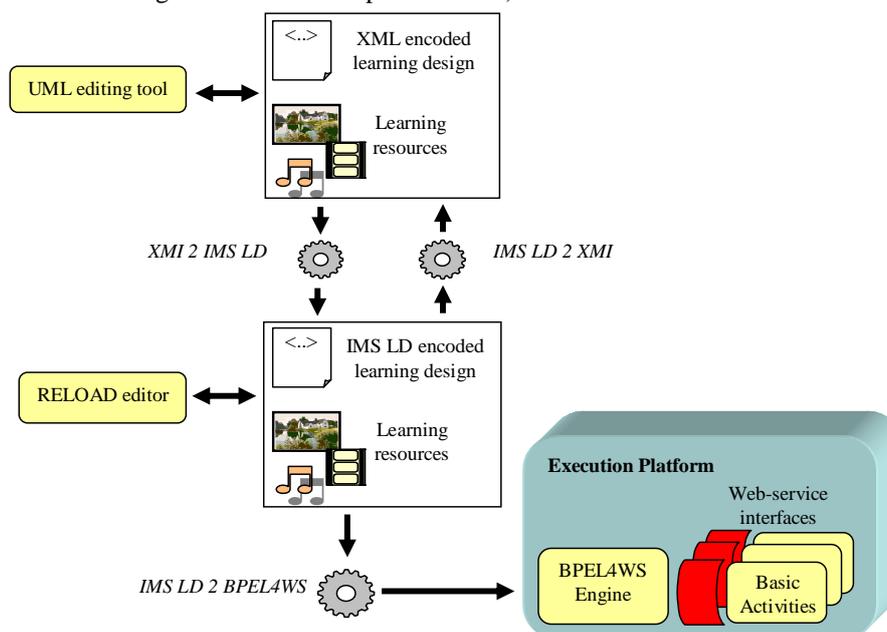

Fig. 4. Integration of IMS LD in <e-LD>

- Operationalization is carried out using a design mapping for translating IMS LD onto BPEL4WS. This mapping, which follows the operationalization conventions described above, is again implemented using an adequate amalgam of XSLT stylesheets with conventional programming support.

5 Related Work

The application of business process techniques as operational support for e-learning solutions and the application of workflow management systems is not new. They have been applied in the creation of LMSs [7], using also BPEL4WS as the workflow language [1], or generalized workflows to support SCORM language [9]. Another interesting work was the ASSIS project [3] in which BPEL4WS was used to orchestrate and integrate an IMS QTI engine and an IMS Simple Sequencing Engine. The main contribution of the present work with respect to these approaches is to promote

a general architecture where educational modeling languages are conceived as the main artifacts for the production and operationalization of e-learning applications. Since these languages can be conceived as particular cases of workflow modeling languages, it is natural to adopt a workflow management system as a common execution platform.

The <e-LD> implementation is service-oriented. Service Oriented Computing and Web Services are becoming standard technologies for the development of software applications. LMSs and other learning tools have been influenced by this emerging technology. Projects like the ELF Framework [6] propose a set of the services that make up an e-learning platform. Another initiative based on web services is the OKI project [12] that defines an e-learning architecture by providing a set of specifications of common services and collections of programming bindings for such services. In addition to these abstract efforts, there are also commercial and open source projects that are based on service oriented computing, an example is the SAKAI project [17]. Finally a more related work to IMS-LD and services is the SleD project [21] which that tries to improve the functionalities of the Coppercore Engine through the integration with services using service oriented computing. In <e-LD> the use of web services is a natural consequence of the choice of BPEL4WS as workflow representation language. The result is an architecture that can be tailored to each specific situation by introducing the appropriate set of supporting services.

6 Conclusions and Future Work

We think that the modelling of the educational processes and their operational support is a key aspect in the construction of more effective e-learning applications. We conceive EMLs as domain-specific languages that can be used directly by educational experts in the modeling of educational processes. In <e-LD> these languages acquire an operational flavor, allowing these experts to lead the development of this kind of applications. The architecture allows the integration of authoring tools with a workflow-oriented execution platform. The connection between these components is carried our using suitable design mappings. Therefore, the architecture can support an open variety of EMLs and authoring styles. In addition, since the execution platform can be enriched with an appropriate set of basic activities, the execution capabilities can be tailored for each specific domain. The main drawback is the effort required to set up the production environment. However, this effort should pay off during the successive production and maintenance stages and it will allow the reuse of workflows tools previously developed for the business domain.

Currently we are finishing the integration in <e-LD> of IMS LD following the approach described in this paper. The next step in the project is to carry out an exhaustive evaluation of the result. We are also working in the development of more integrated authoring support for IMS LD. As a future work we are planning to test other workflow management systems as execution support, as well as to integrate other, more specialized, EMLs.

References

1. Anane, R., B. Bordbar, et al. ,2005, A Web services approach to learning path composition. *Advanced Learning Technologies, ICALT 2005*, IEEE Computer Society.
2. Andrews, T., F. Curbera, et al., 2003, Business Process Execution Language for Web Services version 1.1,.(Retrieved May 18, 2006, http://ifr.sap.com/bpel4ws/)
3. ASSIS Project: http://www.hull.ac.uk/esig/assis.html
4. Coppercore Project, http://www.coppercore.org
5. Dumas, M., van der Aalst, W. M. P., ter Hofstede, A. H., 2005, Process-Aware Information Systems: Bridging People and Software Through Process Technology, Wiley.
6. ELF Framework Project: http://www.elframework.org/
7. Gibson, F. P., 2003, "Supporting Learning in Evolving Dynamic Environments." *Computational & Mathematical Organization Theory* 9(4), pp 305-326.
8. IMS LD, 2003, IMS Learning Design Information Model v1.0 (Retrieved May 1, 2006: http://www.imsproject.org/content/learningdesign/)
9. Kim, K., Yoo, H., and Kim, H., 2005, A Process-Driven e-Learning Content Organization Model. In *Proceedings of the Fourth Annual ACIS international Conference on Computer and information Science (Icis'05)* IEEE Computer Society.
10. Koper, R.; Olivier, B., 2004, Representing the Learning Design of Units of Learning, *Educational Technology & Society*, 7(3), pp. 97-111.
11. Martínez Ortiz, I., Moreno Ger, P., López Moratalla, J., and Fernández-Manjón, B., 2005, Towards the Implementation of IMS Learning Design in the <e-Aula> LMS, *Proceedings of Web-based Education WBE05*, Grindelwald, Switzerland
12. OKI Project: http://www.okiproject.org/
13. Paquette, G., O. Marino, et al. (2005). Implementation and Deployment of the IMS Learning Design Specification. *Canadian Journal of Learning and Technology* 31(2).
14. Polsani, P., 2003, Use and Abuse of Reusable Learning Objects. *Journal of Digital Information*, 3(4), Article No. 164.
15. Rawlings, A. van Rosmalen, P. Koper, R. Rodríguez-Artacho, M. Lefrere, 2002, P. Survey of Educational Modelling Languages (EMLs). CEN/ISSS WS/LT learning technologies workshop. September 19th.
16. Reload Project, website: http://reaload.ac.uk/
17. SAKAI Project http://www.sakaiproject.org
18. Sierra, J. L, Fernández-Valmayor, A y Fernández-Manjón, B. A Document-Oriented Paradigm for the Construction of Content-Intensive Applications. *The Computer Journal. In press,* (doi:10.1093/comjnl/bxl008).
19. Sierra, J.L., et al, 2005, Incremental Definition and Operationalization of Domain-Specific Markup Languages, *ADDS. ACM SIGPLAN Notices*. 40(12) pp28-37.
20. Sierra, J.L., Martinez, I., Moreno, Lopez, J., Fernandez-Manjón, B., 2005, Building Learning Management Systems Using IMS Standards: Architecture of a Manifest Driven Approach. *Lecture Notes in Computer Science* 3583, pp144-156.
21. SLeD: Learning Design Demonstrator, project: http://sled.open.ac.uk/web/
22. Vantroys, T. and Y. Peter, 2003, COW, a Flexible Platform for Enactment of Learning Scenarios. Groupware: Design, Implementation, and Use, *9th International Workshop, CRIWG 2003*, Autrans, France, Springer Verlag.