

## Supporting the Authoring and Operationalization of Educational Modelling Languages

**Iván Martínez-Ortiz**

(Dept. of Software Engineering and Artificial Intelligence  
Complutense University of Madrid, Spain  
imartinez@fdi.ucm.es)

**Pablo Moreno-Ger**

(Dept. of Software Engineering and Artificial Intelligence  
Complutense University of Madrid, Spain  
pablom@fdi.ucm.es)

**José Luis Sierra-Rodríguez**

(Dept. of Software Engineering and Artificial Intelligence  
Complutense University of Madrid, Spain  
jlsierra@fdi.ucm.es)

**Baltasar Fernández-Manjón**

(Dept. of Software Engineering and Artificial Intelligence  
Complutense University of Madrid, Spain  
balta@fdi.ucm.es)

**Abstract:** The modelling of 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 better fit the very particular needs of a learning scenario. In this paper we present <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 workflow management systems, and we describe how this implementation can be used to support IMS Learning Design.

**Keywords:** IMS LD, learning design, graphical notation, units of learning, authoring

**Categories:** K.3.1, K.3.2

### 1 Introduction

In the last years, the representation of learning content in the form of potentially reusable learning objects that can be adapted and assembled in many different e-learning scenarios [Downes 2001, Polsani 2003] has received a lot of attention. However this is only a piece of the whole puzzle. To obtain a full learning experience, the learning process usually involves other activities like solving some problems,

making some experiments in a lab, discussing with the teacher about a specific topic, etc.

Thus, an optimal e-learning application should coherently integrate resources and participants such as students or instructors across a well-defined set of learning activities that are structured carefully and delivered in a learning flow to promote more effective learning. In other words, e-learning applications must be based on a well-founded activity-based educational process.

To achieve this objective, these learning processes must be modelled with the level of detail required in the development of computer applications able to execute them. For this purpose, a suitable *educational modelling language* (EML) can be used [Koper 2001]. An EML is a domain-specific language specially suited for describing instructional processes, which allows for the creation of a formalized version of the learning process that is usually called a *unit of learning* (UoL). A good example of EML is IMS Learning Design (IMS LD) [IMS 2003, Koper and Olivier 2004]. For a comparative study of different EMLs see also [Martínez-Ortiz et al. 2007].

There are several factors that must be addressed regarding the applicability of the EMLs:

- On one hand, the 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 additional objective of making these languages directly executable by computers hinders their usability. For instance, to have educators who directly represent their educational designs in the XML encoding of IMS LD is not a realistic assumption, even with the help of XML editing tools (e.g. XML Spy). For this purpose, authoring tools must be provided.
- On the other hand EMLs are equipped with operational semantics, and therefore they can be interpreted by execution platforms in order to automatically produce e-learning applications. Moreover, there is no universal EML that can be applied to all learning scenarios. In fact, there are different types of EMLs that can be catalogued in terms of their complexity and objectives. In addition, while the use of a standard EML (e.g. IMS LD) makes it possible to reuse educational designs (i.e. UoLs) and tools (e.g. execution platforms and players), the EML itself can evolve into 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, it is that there is no such thing as a universal solution). 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 flexibility required by the effective use of EMLs in realistic learning scenarios. It is a revised and extended version of the previous work described in [Martínez-Ortiz et al. 2006].

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 [Figure 1]. This architecture, which is conceived to accommodate the different sources of variability identified in the introduction, distinguishes the following components:

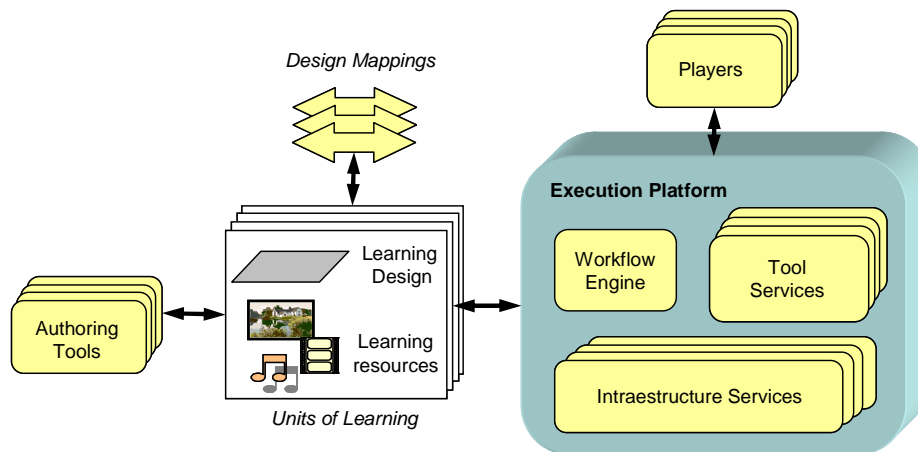


Figure 1: The <e-LD> conceptual architecture

- *Units of learning.* The main goal of the architecture is to produce e-learning applications based on units of learning. 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 (e.g. contents, services, etc.). Notice that, although the terminology used is to some extent borrowed from IMS LD, here we are using it in a conceptual and a representation-agnostic sense.
- *Authoring tools.* These tools are used by instructional designers to formalize the learning designs that are integrated in the units of learning. Besides the educational concepts, EMLs usually include technical concepts or concepts related to reusability and standardization efforts (e.g. LOM Metadata, IMS Content Packaging). However, these elements mostly distract the attention of the designer from her main task: producing good learning designs. Hence, one of the objectives of authoring tools must be to isolate learning designers from those aspects that are not directly relevant for their needs. A good example of an authoring tool is RELOAD Editor for IMS LD (web site: [www.reload.ac.uk](http://www.reload.ac.uk)).
- *Execution Platform.* This component is intended to help during the execution of the learning process, coordinating the people involved in this process and

also providing tools and learning contents. This component plays a role analogous to engines for IMS LD such as Coppercore (web site: [www.coppercore.org](http://www.coppercore.org)). As recognized by some researchers [Vantroys and Peter 2003, Paquette, Marino et al. 2005] and in the IMS LD specification itself, learning designs can be viewed as particular cases of workflows, as defined in the context of business process integration [Dumas, Aalst et al. 2005]. 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. Moreover this component includes different support services (e.g. a dossier service to manage user model properties) that are internal to the execution platform, usually used for management purposes. In addition, this component includes definitions of the different tools that will be used during the running of the unit of learning. The implementation of these services can be packaged inside the execution platform or can be implemented outside (e.g. adapting an existing application to the tool service contract).

- *Players*. These components are the graphical user interfaces that are used by students and teachers to interact with a unit of learning during its execution. Therefore they have a similar role to IMS LD players such as IMS LD Reload player or SLeD (a player for Coppercore; web site: [sled.open.ac.uk](http://sled.open.ac.uk)). Players also interact with tools and other services used, providing a unique user interface for all of these tools.
- *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, authoring tools can be designed in an EML-neutral way and later on translated to a suitable specific EML. In addition, those mappings are used to translate a UoL into a specific EML, or into a general-purpose execution-oriented format (e.g. a workflow-oriented language script). It is interesting to notice that a design mapping architecture does not require these mappings to be entirely automatic. Instead, they can comprise intervention by the user (e.g. in order to translate complex parts). 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 experimental XML-based <e-Aula> Learning Management System [Sierra, Martínez-Ortiz et al. 2005, Sierra, Moreno-Ger et al. 2007].

To facilitate the authoring of units of learning, a graphical notation for the concepts typically found in an activity-oriented EML is proposed. This graphical

notation is intended to be used not only during the authoring process but also to facilitate the reuse of already available units of learning. In particular, the UML-like notation [OMG 2005, OMG 2006] has been selected.

Regarding the operationalization of EMLs, we have chosen BPEL4WS [Andrews, Curbera et al. 2003] as the workflow language. The resulting implementation is outlined in [Figure 2]. BPEL4WS plays the role of conductor, coordinating learning activities and tools that will be used during the learning process.

In this implementation, learning designs must be represented using suitable XML-based domain-specific languages. Notice that this is not a severe constraint, since XML is currently being adopted as a standard interchange format between tools, and existing non XML-compliant authoring tools can be wrapped to produce an XML-based representation. Besides, it is also common practice for the existing EMLs to provide an XML binding, and it also facilitates the provision of EMLs tailored to specialized domains following a document-oriented approach, such as those described in [Sierra, Fernández-Valmayor et al. 2004, Sierra, Fernández-Manjón et al. 2005, Sierra, Fernández-Valmayor et al. 2006]. Finally, BPEL4WS itself also has an XML-based language.

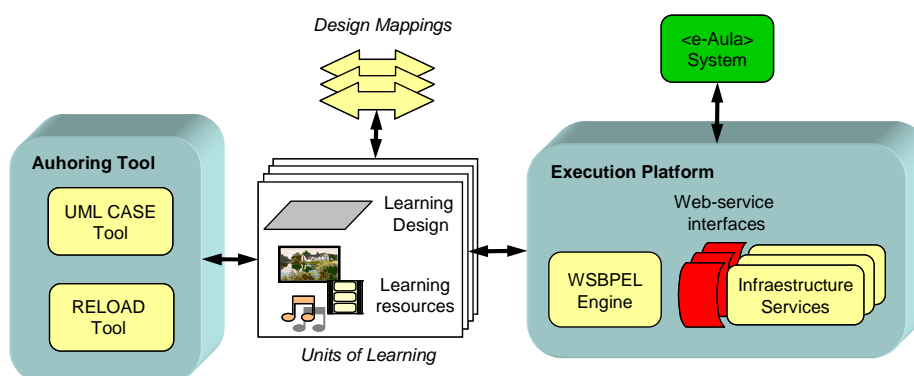


Figure 2: <e-LD> BPEL4WS-based implementation

As a consequence of the use of XML, the design mappings are transformations 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 such as XSLT. For more complex mappings, it may 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 [Sierra, Navarro et al. 2005]. 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.









