Service Oriented Architecture to Manage Units of Learning based on the IMS LD specification

Juan C. Vidal[1], Adrian Novegil[1], Manuel Lama[1] and Eduardo Sánchez[1]

[1] Department of Eletronics and Computer Science.
University of Santiago de Compostela.
jvidal@dec.usc.es, adrian.novegil@rai.usc.es, lama@dec.usc.es, eduardos@usc.es.

Keywords
IMS Learning Design; Service Oriented Architecture; Ontologies; Petri Nets.

1. EXECUTIVE SUMMARY

A learning design describes the method that enables learners to achieve learning objectives after carrying out a set of activities using the resources of an environment. Among the current proposals, the IMS Learning Design (IMS LD) (IMS Consortium, 2003) specification has emerged as the de facto standard that facilitates the representation of learning designs independently of the chosen pedagogical focus.

Nowadays, the *CopperCore* engine (Vogten and Martens, 2005) is the most used platform available for executing IMS LD learning designs. This engine has two main drawbacks, which are related with its reuse and maintenance: (1) its design does not facilitate its integration with other e-learning applications; and, (2) if the IMS LD is either modified or evolved in order to include new requirements, the current engine as well as the e-learning application in which it could be integrated, should be re-designed and re-implemented.

To overcome these limitations, we have developed a service-oriented architecture based on web service technologies that allows developers to design and execute units of learning following the IMS LD specification. The main features of this architecture are described:

- The learning flow model of each unit of learning is represented through a Petri net model that describes the execution coordination of the learning design components (plays, acts and activities). The architecture includes services that **(1)** automatically translate the IMS LD learning flows into the Petri net model; and **(2)** carry out the tasks needed to manage the execution of the Petri net, or equivalently, of the learning design.
- IMS LD units of learning are semantically described with an ontology (Amorim et al., 2006) that enables developers to validate the learning design. In this ontology, the IMS LD elements are modeled in a concept taxonomy in which the relations between the concepts are explicitly represented. In addition, a set of axioms constraining the semantics of the concepts has been formulated from the restrictions (expressed in natural language) identified in the analysis of the IMD LD specification.
- The architecture includes services that **(1)** translate from specific learning designs into ontology instances, and **(2)** use an ontology reasoner, like FLORA2 (Yang et al., 2003), to guarantee that units of learning are correctly represented following the semantics of the IMS LD.

With the proposed engine, two main benefits can be outlined: (1) the Petri net model can easily be reused to support different e-learning specifications by means of changing the translation services; and (2) the authoring component is decoupled from the execution one which opens the door to execute a Unit of Learning in different engines, each one executing different specifications.

2. INTRODUCTION

Last decade, a large number of tools for the creation, publication and execution of units of learning have appeared (Moodle, 2008; Sakai, 2008; Escobedo et al., 2007; LAMS, 2008). These tools facilitate student's access to learning contents and activities and are usually based on Educational Modeling Languages (EML) (Rawlings et al.. 2002; Koper, 2001). EML aims for describing *from a pedagogic point of view* the learning design of a course: that is, the flow of the learning activities carried out by the students to achieve the objectives of a course using a given educational content. Among the current proposals, the IMS Learning Design (IMS LD) (IMS Consortium, 2003) specification has emerged as the de facto standard that facilitates the representation of learning designs independently of the chosen pedagogical focus. Therefore, new tool developments for the publication and execution of units of learning must provide or use an engine that can interpret the semantics of the IMS LD specification.

At present, there are several engines for the execution of units of learning based on the IMS LD specification (Vogten and Martens, 2005; Escobedo et al., 2007; Hagen et al., 2006) that share two main drawbacks, which are related with reuse and maintenance: (1) their design and implementation do not facilitate their integration with other e-learning applications because they do not provide the methods for a remote invocation of the execution of the learning designs; and, (2) if the IMS LD is either modified or has evolved to include new requirements, the engine as well as the e-learning application, in which it could be integrated, should be re-designed and re-implemented. It is because these tools only process the XML-Schema language of the IMS LD and usually do not take care of the tasks coordination during the courses.

To overcome these limitations, we have developed a service-oriented architecture based on web service technologies that allows developers to design and execute units of learning following the IMS LD specification. The main features of this architecture are:

- The learning flow model of each unit of learning is represented through a Petri net (Murata, 1989) model that describes the execution and coordination of the learning design components (plays, acts and activities). The architecture includes services that **(1)** automatically translate the IMS LD learning flows into the Petri net model; and **(2)** carry out the tasks needed to manage the execution of the Petri net, or equivalently, of the learning design. Therefore, a unit of learning execution is based on the Petri net that models the IMS LD components and not in an interpretation of the IMS LD specification. Note that if the IMS LD specification is modified, the changes only affect the Petri net translation of the learning design but not the architecture that executes it .
- IMS LD units of learning are semantically described with an ontology (Amorim et al., 2006) that enables developers to validate the learning design. In this ontology, the IMS LD elements are modeled in a concept taxonomy in which the relations between the concepts are explicitly represented. In addition, a set of axioms constraining the semantics of the concepts has been formulated from the restrictions (expressed in natural language) identified in the analysis of the IMD LD specification.
- The architecture includes services that **(1)** translate from specific learning designs into ontology instances, and **(2)** use an ontology reasoner, like FLORA2 (Yang et al., 2003), to guarantee that units of learning are correctly represented following the semantics of the IMS LD.

With the proposed engine, two main benefits can be outlined: (1) the Petri net model can easily be reused to support different e-learning specifications by means of changing the translation services; and (2) the authoring component is decoupled from the execution one which opens the door to execute a Unit of Learning in different engines, each one executing different specifications.

The paper is structured as follows: Section 3 introduces the learning design specification and the learning execution problem this paper is focused on. Section 4 presents the architecture that supports this execution while Section 5 describes the services this architecture provides. Finally, Section 6 points out the conclusions of this work.

3. IMS LEARNING DESIGN SPECIFICATION

The IMS LD specification, drawn up by the IMS/LDWG work group, is an integration of the EML developed by the OUNL (Open University of Netherlands), with other existing IMS specifications aimed at the exchange and interoperability of e-learning material. IMS LD (**¡Error! No se encuentra**

el origen de la referencia., 2003; IMS, 2006) describes a method that is made up of a number of activities carried out by both learner and staff in order to achieve some learning objectives. It allows the combination of various techniques (traditional, collaborative, etc.), and facilitates the description of new ones. From the proposed specifications, the IMS LD has emerged as the de facto standard for the representation of any learning design that can be based on a wide range of pedagogical techniques.

The IMS LD specification has been formally modeled through the XML-Schema language (Thompson et al, 2004). However, the knowledge model of this language is not expressive enough to explicitly define (1) hierarchical (is-a) relations between two or more concepts, (2) properties of relations, and (3) general and formal constraints (axioms) between concepts, attributes, and relations.

These knowledge representation issues are solved with a IMS LD ontology (Amorim et al, 2006). The IMS LD elements are modeled in a concept taxonomy in which the relations between the concepts are explicitly represented. In addition, a set of axioms constraining the semantics of the concepts has been formulated from the restrictions (expressed in natural language) identified in the analysis of the IMD LD specification. On one hand, as the semantics of the concepts is precisely defined, there should be no misinterpretations or errors when the instances of the concepts are created and managed in runtime. In this sense, new concepts, attributes/relations, and formal axioms have been identified and formalized in the ontology. It is important to emphasize that these add-ons neither change nor extend the IMS LD specification, but they enrich the description of the semantics of the IMS LD elements. On the other hand, as the semantics of the IMS LD specification is explicitly described, it is not necessary to codify such semantics in the development of the software program that allow to users to design and execute the unit of learning. Thus, a general reasoner, following the logic paradigm associated to the language in which the ontology is represented, can be used to check the consistence of the unit of learning in both the design and runtime phases. To summarize, the IMS LD ontology can be used on the following stages:

1. Authoring Stage. A set of axioms could be used by authoring agents to guide the creation of IMS LD documents, as well as to validate their logical consistency (Sánchez et al, 2007).

2. Execution Stage. The ontology could be used by production agents to automate different tasks such as: resource allocation based on availability, creation of new IMS LD runs based on the number of registered students, role assignments based on student profiles, and so on. Furthermore, the ontology could also be used in the delivery process to solve the problem of personalization or adaptation of the learning design to the individual users.

This paper is focused on the execution stage. By means of the IMS LD ontology, IMS LD elements can be translated into IMS LD ontology instances, and then ontology reasoners can be used both to validate and manage the execution of Units of Learning.

In the following section, the translation from IMS LD ontology concepts into Petri Net elements as well as the execution of those nets in a SOA-based execution engine, are explained.

4. SYSTEM ARCHITECTURE

Increasingly organizations rely on Service Oriented Architecture (SOA) (Erl, 2007) to increase their productivity, improve their operational efficiency and speed of reaction, as well as to align their infrastructure with business strategies. Unfortunately the present SOA scene is dotted with trendy neologisms, technologies in conflict and without taking into account that there is no single approach or single solution for the same problem.

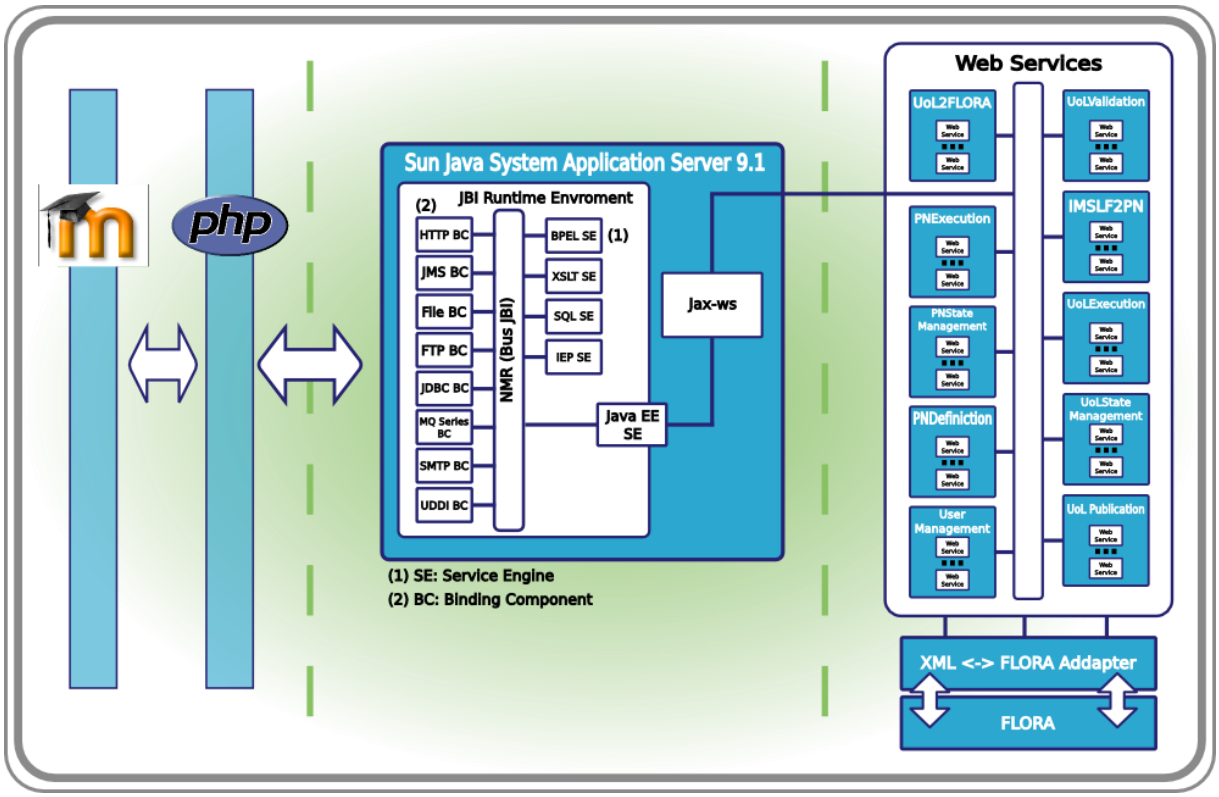

Figure 1: Three-layered service oriented architecture that manages the units of learning

Figure 1 represents a three-layered system architecture developed following the design principles of SOA:

- The first layer represents the e-learning tool that both students and teachers use to access to the courses and therefore to their content, design and edition. It is based on the e-learning tool Moodle (Rice, 2008, Moodle, 2008) which is a free distribution course management system (CMS) that helps educators to create online learning communities. In addition, Moodle provides an invocation layer for web services that is written in PHP and is in charge of the dialogue with the architecture. This sub-level is also performs the invocation and binding of data, making client and server as loosely coupled as possible. This layer corresponds to Moodle and PHP boxes of Figure 1.

- The second layer contains the core and interconnection elements of our architecture: OpenESB (OpenESB, 2008). OpenESB or Open Enterprise Service Bus (Chappell, 2004) is an implementation of an Enterprise Service Bus-based JBI (Ten-Hove, 2005) developed by Sun Microsystems (Figure 2). This bus facilitates an easy integration of Web services and enabling the creation of loosely coupled applications. To develop this type of applications, Open Enterprise Service Bus (Open ESB) hosts a set of pluggable component containers, which integrate various types of IT assets. These pluggable component containers are interconnected with a fast, reliable, and in-memory messaging bus called the Normalized Message Router (NMR) also referred to as the JBI Bus. Service containers adapt IT assets to a standard services model, based on XML message exchange using standardized message exchange patterns (MEP) based on abstract WSDL. This improves interoperability and allows a mix-and-match of technologies from various vendors.

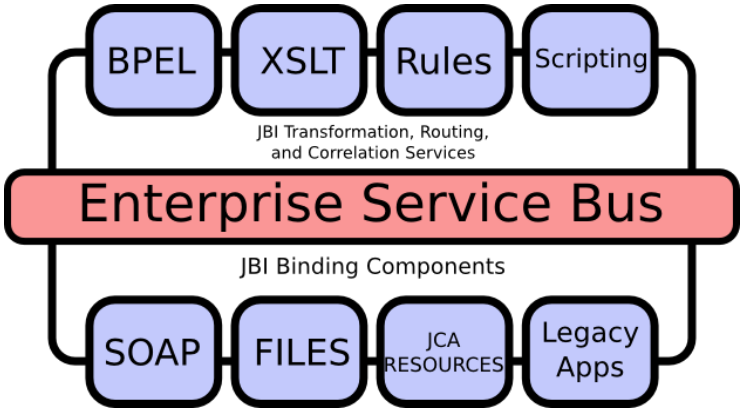

Figure 2: The core of the architecture is an enterprise service bus. This bus facilitates a loosely coupled integration of applications through Web services.

- The last layer represents the set of Web services (Chappell, 2002, Hansen, 2007) that implement the business logic of our application. In this case, the implementation of services is based on the Web Services Stack developed by Sun Microsystems and named Metro (Metro, 2008). Metro is a web service stack that allows an easy and secure development and deployment of web services. In essence Metro implements all the necessary tools to work with Web services: security, reliability, transactions, as well as the compatibility with .NET services. The Metro web service stack is part of the GlassFish community, but it can be also used outside GlassFish. This layer also contains the reasoner FLORA-2 (Flora-2, 2008, Yang et al. 2003) which is an advanced object-oriented knowledge base language and application development environment. The language of FLORA-2 is a dialect of F-logic with numerous extensions, including meta-programming in the style of HiLog and logical updates in the style of Transaction Logic. This reasoner provides the means for reasoning with ontologies. In this work, we extended this reasoner with a layer in F-Logic which allows us to validate and reason with units of learning and Petri nets.

5. SYSTEM FUNCTIONALITIES

The Web services layer of the architecture depicted in Figure 1 provides the following set of services:

- UoL2Flora: Set of services that translate the units of learning from IMS XML-based files to the ontology of units of learning and so to the underlying reasoner Flora-2. A Java-based middleware handles this translation and also manages the persistence.
- UoLValidation: Set of services for the validation of units of learning. The ontology of units of learning provides the set of axioms that enable this validation. These services ask the reasoner for errors or inconsistencies in the unit of learning design. Errors are returned to the user indicating the ID of the element that violates the axiom, error description, and possible causes.

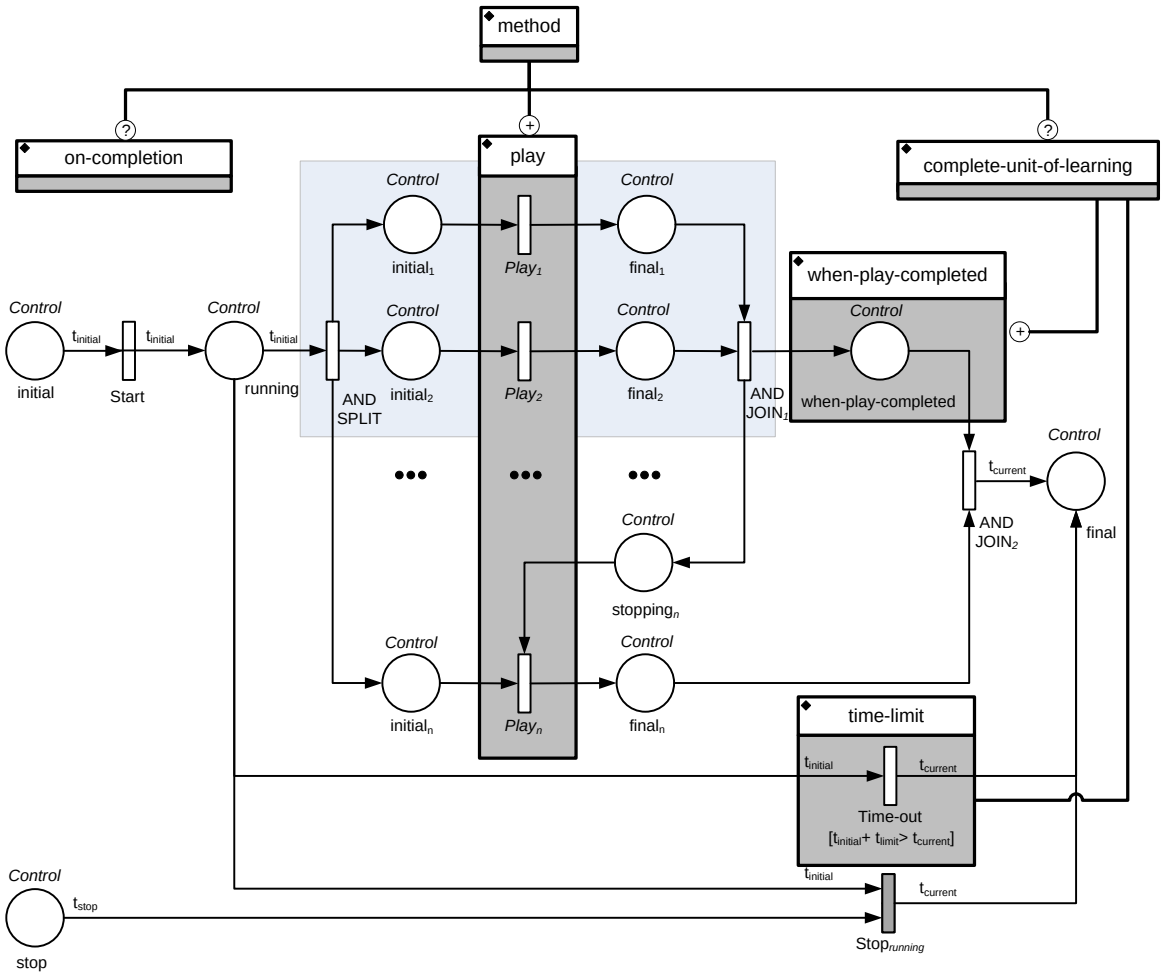

Figure 3: Petri net structure for the execution of a method. This net is decorated with the method definition of the IMS specification so we can see the mappings between both models

- UoLExecution: Set of services that allow seeking enforcement of a unit of learning. These services retrieve a unit of learning instance that a user is searching for. Specifically, they recover the specific unit of learning instance from the Petri nets repository and load it into the executor.
- UoLStateManagement: Set of services that manage the state of a Petri net. They allow us to query the state of the network as to stop or restart it. A network may be in the following states: ready, paused, suspended or running.
- UoLPublication: Set of services that allow the publication of units of learning in the system. Authorized users may publish new units of learning making these units accessible to other users. Specifically, these services make the translation and the validation of the unity of learning and finally insert it in the database and the executor.
- UserManagement: Set of services for user management. A user can manage his profile, roles, courses he is taking and so on through these services.
- Besides the functionalities previously exposed, the system has a collection of web services that enable the management of the Petri nets that model the units of learning. The execution of a unit of learning can be seen as the coordination of a set of activities with a set of participants. If we consider that these activities are processes, we can reduce this problem to the execution of a workflow. For this reason, this work approaches the execution of a unit of learning as a workflow modeling problem where the learning and support activities of a learning design will represent the tasks to perform whereas the methods,

Figure 4: Sequence diagram that models the logic of the publication of a unit of learning

plays and acts will constraint the workflow structure. Figure 3 depicts the Petri net that represents the execution of a method. Moreover, this net also shows how the *method* concept defined in the IMS specification is mapped into the Petri net, that is, how each of its attributes (with a dynamic behavior) is related to a place or transition of the Petri net. That is, it related *(i)* the set of plays with the set of concurrent *Playi* transitions, and the two conditions that specify when the method is completed: *(ii)* associating the set of plays that must be completed with the *when-play-completed* place that synchronizes their execution, and *(ii)* defining the *time-limit* constant in the *Time-out* transition.

- IMS2PN: This service is responsible for converting a unit of learning to a Petri net. This translation is done automatically because each unit of learning has a Petri net that models its behavior. Petri nets are also modeled through an ontology (Vidal et al., 2006).
- PNExecution: Set of services that enable the management of the Petri nets. These services retrieve Petri nets from the repository and load them into the executor.
- PNStateManagement: Set of services that enable the management of the state of a Petri net. Like units of learning, there are services to start, stop or reset Petri nets execution.
- PNDefinition: Set of services that allow the definition a Petri net graph, that is, the definition of the net nodes, arcs, signature, terms and so on. These services facilitate the Petri net-based definition of units of learning.

For example, Figure 4 shows the sequence of events for the publication of a unit of learning. When a user that is registered in the system requests the publication of a new unit of learning, the system invokes the publishing service with the user ID and the unit of learning file as parameters. Associated with this publishing service, the system has a BPEL process that is responsible for the orchestration of the various services that are involved in the operation. For example, the system first invokes the user authentication and authorization service, and if the user has the privileges then it invokes the service validation of the unit of learning. If the validation is negative, the service is interrupted and the errors are returned to the user via an error message. On the contrary, if it is valid, the service translates the unit of learning from IMS format to the ontology used by the execution engine so this unit is available to other users.

6. CONCLUSIONS

The Service Oriented Architecture presented in this work facilitates the publication and execution of learning designs based on the IMS LD specification. IMS LD components are modeled by means of Petri nets and thus its execution follows the execution rules of this formalism. This solution makes IMS LD specification independent from its execution and therefore reduces the effects that changes in the IMS LD specification might cause in the execution engine.

On the other hand, the services this architecture provides are based on ontologies that describe both the IMS LD specification as well as the Petri nets. This ontological orientation facilitates *(i)* a declarative translation between the IMS LD and Petri net models and *(ii)* the validation of units of learning since the own ontologies contain the axioms that restrict the semantics of the models.

7. ACKNOWLEDGEMENTS

Authors wish to thank the Xunta de Galicia and the Ministerio de Educación y Ciencia for their financial support under the projects PGIDIT06SIN20601PR and TSI2007-65677C02-01.

8. REFERENCES

Vogten, H., and Martens, H. (2005). *CopperCore 2.2.2*, Located on http://www.coppercore.org.

IMS Consortium (2003). *IMS Learning Design Information Model. Version 1.0 Final Specification*, Retrieved from: http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html.

Amorim, R., Lama, M., Sánchez, E., Riera, A. and Vila, X.A. (2006). A Learning Design Ontology based on the IMS Specification. IEEE Journal of Educational Technology Society, volume 9, number 1, pages 38-57.

Vidal, J. C., Lama, M., & Bugarín, A. (2006). A High-level Petri Net Ontology Compatible with PNML. Petri Net Newsletter, volume 71, pages 11-23.

Yang, G., Kifer, M., & Zhao, C. (2003), Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. Proceedings of the Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), Catania, Sicily, Italy, pages 671-688.

Flora-2 website (2008). Retrieved from: http://flora.sourceforge.net.

Rice, W. (2008). Moodle 1.9 E-Learning Course Development. PACKT publishing.

Moodle website (2008). Retrieved from: http://moodle.org.

Sakai website (2008). Retrieved from: http://www.proyectosakai.org.

LAMS website (2008). Retrieved from: http://www.lamsinternational.com.

Erl, T. (2007). SOA Principles of Service Design. Upper Saddle River, NJ, USA: Prentice Hall PTR.

OpenESB website (2008). Retrieved from https://open-esb.dev.java.net.

Ten-Hove, R., & Walker, P. (2005). Java Business Integration (JBI) 1.0. Sun Microsystems, Inc. Retrieved from http://jcp.org/en/jsr/detail?id=208.

Chappell, D. (2004). Enterprise Service Bus. Sebastopol, CA, USA: O'Reilly Media, Inc.

Hansen, M. D. (2007). SOA Using Java(TM) Web Services. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Metro website (2008). Retrieved from: https://metro.dev.java.net.

Chappell, D., & Jewell, T. (2002). Java Web Services. Sebastopol, CA, USA: O'Reilly Media, Inc.

IMS Consortium (2006). IMS Learning Design Best Practice and Implementation Guide. Retrieved from: http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html.

Thompson, H., Beech, D., Maloney, M., & Mendelsohn, N. (2004). XML-Schema Part 1: Structures Second Edition, retrieved May 20, 2005 from http://www.w3.org/TR/xmlschema-1

Eduardo Sánchez, Manuel Lama, Ricardo R. Amorim and Angel Negrete. WebLD: A Web Portal to Design IMS LD Units of Learning. Proceedings of the ICALT´07 Conference, 2007.

Escobedo del Cid, J. P. de la Fuente Valentin, L., Gutierrez, S., Pardo, A., & Delgado Kloos, C. (2007). Implementation of a Learning Design Run-Time Environment for the .LRN Learning Management System. Journal of Interactive Media in Education, volume 1, pages 1-12.

Rawlings, A., Rosmalen, P., Koper, R., Rodríguez-Artacho, M., & Lefrere, P. (2002). Survey of Educational Modelling Languages. CEN/ISSS WS/LT Learning Techonologies Workshop.

Koper, R. (2001). Modelling units of study from a pedagogical perspective the pedagogical meta-model behind EML. Retrieved from: http://eml.ou.nl/introduction/docs/ped-metamodel.pdf.

Hagen, K. and Hibbert, D. and Kinshuk, P. (2006). Developing a Learning Management System Based on the IMS Learning Design Specification. Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, Kerkrade, The Netherlands, pages 420-424.

Murata, T. (1989). Petri nets: Properties, Analysis and Applications. Proceedings of the IEEE, volume 77, number 4, pages 541-580.