Torres, J., Resendiz, J., Aedo, I., Dodero, J.M. (2014). A model-driven development approach for learning design using the LPCEL Editor. *Journal of King Saud University-Computer and Information Sciences*, **26**(1) supplement, pp. 17-27.

DOI: 10.1016/j.jksuci.2013.10.004

Journal of King Saud University – Computer and Information Sciences (2014) **26**, 17–27

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

A model-driven development approach for learning design using the LPCEL Editor

Jorge Torres [a,*], Jesús Resendiz [a], Ignacio Aedo [b], Juan Manuel Dodero [c]

[a] *Departamento de Computación, Tecnológico de Monterrey, Querétaro, Mexico*
[b] *Departamento de Informática, Universidad Carlos III de Madrid, Madrid, Spain*
[c] *Escuela Superior de Ingeniería, Universidad de Cádiz, Cádiz, Spain*

Available online 22 October 2013

KEYWORDS

Learning design;
Web services;
Model-driven development;
Learning patterns;
Collaboration

Abstract Complex learning scenarios are represented using Educational Modeling Languages (EMLs). Different people with specific skills usually design these scenarios. The IMS LD is a commonly used EML for which some visual editors are being created in order to assist the authoring process. However, these editors have a limited level of expressiveness and do not provide the facilities for designers to collaborate in the design process. The LPCEL Editor provides a broad level of expressiveness and facilitates the authoring process with an editor that includes: (1) Visual Elements, (2) Intermediate Representation, (3) Learning Patterns, (4) Collaboration tools and (5) Web Services. In order to verify that the components are sufficient, we conducted a user evaluation to analyze their perspective regarding the level of functionality of the tools.

1. Introduction

Learning Design is concerned with the creation and delivery of activities and resources that represent a learning scenario in which a learning experience takes place. Educational Modeling Languages (EMLs) provide the framework to design learning scenarios, including the tools needed to represent the activities, resources, services, and user roles (Koper, 2001). EMLs are intended to represent Complex Learning Processes (CLP) by means of the dynamic integration of teaching methods and resources, based on the collaboration between instructors and students in a learning process (Dodero et al., 2010). A Unit of Learning (UoL) is the smallest unit of work of a CLP and is formed by a set of activities that students need to complete in order to achieve specific learning objectives.

The IMS LD (Koper and Tattersall, 2005) is a commonly used EML designed to fulfill a wide spectrum of learning situations (Torres et al., 2012). However, some situations cannot be properly represented with IMS LD due to its level of expressiveness (Torres et al., 2009a) and the lack of functionality of the authoring tools (Martínez-Ortiz et al., 2009). To overcome the former issues, the Learning Process Composition and Execution Language (LPCEL) is proposed. The LPCEL (Torres et al., 2006) describes complex learning scenarios using dynamic flow structures, resources and services, as well as the definition of the execution stage for the learning scenario. The objective of this paper is to describe the components included in the LPCEL Editor and their capabilities to facilitate different aspects of the authoring process and to define and

* Corresponding author.
E-mail addresses: jtorresj@itesm.mx (J. Torres), jresenzp@itesm.mx (J. Resendiz), aedo@ia.uc3m.es (I. Aedo), juanma.dodero@uca.es (J.M. Dodero).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

1319-1578 © 2013 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jksuci.2013.10.004

deploy effective e-learning scenarios that must be seamlessly integrated with the overall design of the CLP.

Representing learning scenarios requires a deep understanding of the low-level details of a given EML and is more complicated when using its advanced functionality (Martínez-Ortiz et al., 2008). The IMS LD has three levels to describe a learning scenario: Level A describes the primitives that represent the static part of the course (activities, roles, resources, etc.). Level B adds properties that allow some degree of dynamism for the learning flow when certain conditions are met. Level C provides the ability to use notifications for the level B to enable certain level of dynamism.

To prevent users from having to use XML elements for an EML, visual authoring editors for Learning Design have been created. They provide a set of tools and iconographic representations of the primitives of an EML (Paquette et al., 2006). Some visual tools are IMS LD compliant so they are also restricted by the level of expressiveness provided by the IMSLD.

Although visual editors have representations of the primitives for some EMLs, they also include other mechanisms that offer a better support for the authoring process, e.g. the collaborative design of learning objects (Padrón et al., 2005), which involves several users in the creation of complex scenarios. There are several approaches to create learning patterns for the IMSLD specification, particularly for collaborative learning (Hernández-Leo et al., 2004). The objective of the collaborative learning patterns is to reuse the solutions found for certain problems during the design of a course. Generic service-based approaches (De la Fuente et al., 2008) can be used to integrate third party web services in the CLP. However, this task is complicated because it requires knowledge about the protocol in which the service is being described.

2. Related work

Creating learning scenarios requires users with a deep knowledge about the low-level details (XML tags and structure) of the EML they are using to describe the learning process. However, it is recommended for them to use a tool that facilitates the authoring of CLPs via graphical abstractions for the low-level primitives of a given EML (Martínez-Ortiz et al., 2008).

A number of visual authoring tools have simplified the design of CLPs, which are delivered often as IMS LD-compliant UoLs. The ReCourse editor (Griffiths et al., 2009) can display the flow of activities of a UoL, but it has no visual support for the level B of the IMS LD specification. CompendiumLD (Atkinson and Kuhne, 2003) resembles a mind-mapping tool where sequences of tasks can be designed, but it does not translate designs to any standard EML. Other tools such as MOT+ (Paquette et al., 2005) and ASK LDT (Sampson et al., 2005) enable to visually edit IMS LD level A and B compliant UoLs. The UML4LD (Laforcade, 2007) can reverse-engineer an IMS-LD compliant XML file but it does not allow editing or creating learning scenarios. The FlexoLD visual editor (Dodero et al., 2010) manages the IMS LD level B properties through high-level abstractions (e.g. branches and loops) that generate a UoL ready to run in an IMS LD engine or in a specific learning scenario, but has no visual facilities for editing the web service interactions. The COLLAGE tool (Hernández-Leo et al., 2004) provides patterns for collaborative learning environments, but the creation or modification of patterns is complicated since it requires programing skills. The LAMS system (Dalziel, 2003) provides a group of tools to design, manage and deploy learning scenarios, using its own graphical representation and EML, but it does not include mechanisms for administrating patterns or web services.

These tools are useful for designing learning scenarios, but they do not necessarily facilitate the process where several users are working at the same time (as it normally occurs). The CASLO infrastructure (Padrón et al., 2005) provides a version control system to manage the changes done in the learning scenario and includes also an agreement system; however, the collaboration is done through an XML file. The Odyssey VCS (Murta et al., 2008) provides a UML versioning system where models are updated automatically when a change is made, but it does not give a complex management of users (roles and permissions).

Because of the limitations of these approaches, a tool that can describe a more expressive EML is needed so the representation of CLPs can be facilitated. The tool must be based on an expressive language, such as the LPCEL.

3. Design of virtual learning experiences

The authoring process of a Virtual Learning Experience (VLE) can be done using visual instructional design languages (Caeiro Rodríguez et al., 2010). The LPCEL Editor not only provides visual components for the authoring process, but it also incorporates other elements that support the authoring of VLEs. The architecture of the LPCEL Editor has five main components as depicted in Fig. 1: (1) The Visual Editor, (2) Intermediate Representation, (3) Learning Patterns, (4) The Collaboration scheme and (5) Use of web services. The LPCEL Editor has been described before (Torres et al., 2012), however this work presents an improvement over the last version although some similarities can be found between these publications.

3.1. Visual editor

Model Driven Development (MDD) is a software engineering approach consisting of techniques to produce models rather than computer programs (Atkinson and Kuhne, 2003). The MDD uses models that are less bound to a target implementation and are closer to the problem and its specific domain, enabling users to focus on solving problems rather than implementation issues. The Model-Driven Learning Design (MDLD) Dodero et al. (2012) proposes a set of visual abstractions for the authoring process and the mechanisms to generate XML files compliant with an EML.

The LPCEL Editor uses a subset of the Business Process Modeling Notation (BPMN) 2.0 Murta et al. (2008) as the visual layer to create an abstraction for the LPCEL primitives. The BPMN is a graph-oriented language where the flow of nodes can be specified in a simple fashion. The subset used by the LPCEL Editor can be divided into three groups as depicted in Fig. 2: (1) Basic Elements, (2) Complex elements and (3) Artefacts.

Group number one is composed by the (1) Basic Activity, (2) Start, (3) End, (4) Gateway, (5) Intermediate and the (6) Variable; they are used to represent simple tasks and the flow

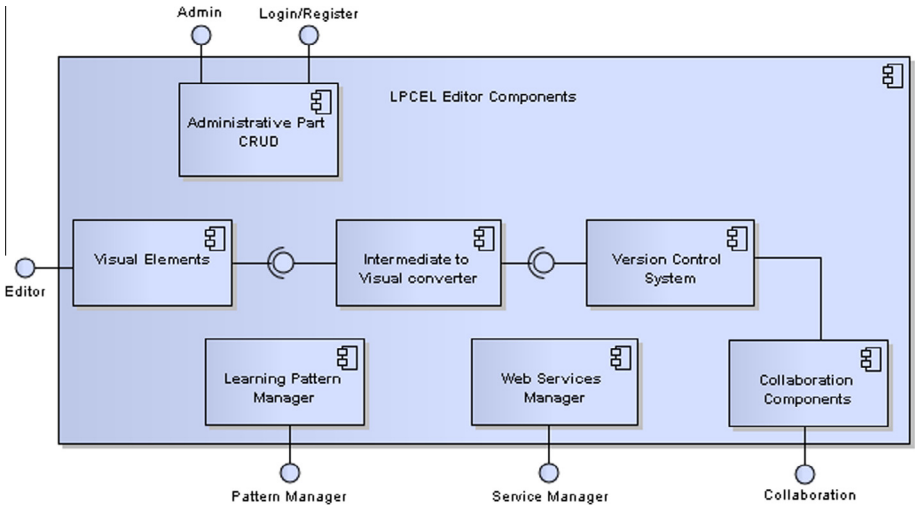

Figure 1 LPCEL editor architecture.

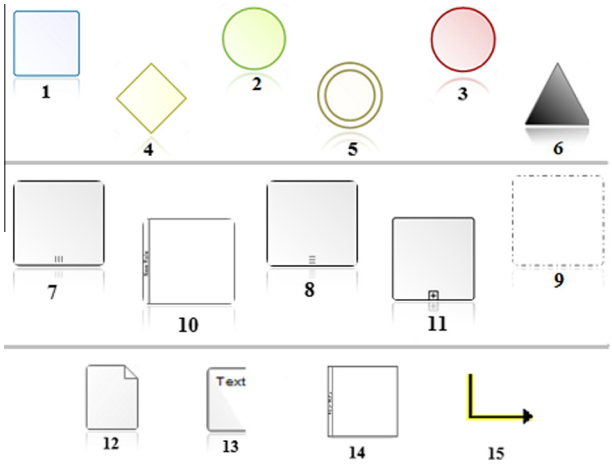

Figure 2 Visual elements of the LPCEL editor.

control for a learning process. The second group is composed by the (7) Sequence Activity, (8) Parallel Activity, (9) Group, (10) Role and the (11) Subprocess elements; they are used to represent the Complex Learning Process of a course, matching the high level of expressiveness provided by the LPCEL. The last group is composed by the (12) Data Object, (13) Text Annotation, (14) Lanes and (15) Connectors; that provide specific behavior for the VLE.

Using these elements, the user can be more focused on the pedagogical intention and the learning objectives of a given learning course, without being aware of the XML implementation details for the same scenario, nor the appropriate XML structure.

3.2. Intermediate representation

The Intermediate representation language for the LPCEL visual abstraction is created in plain text using the JavaScript Object Notation (JSON) (Crockford, 2006) which is a lightweight, text-based, human-readable, language-independent data interchange format. The JSON structure created to save the selected state of the visual abstraction is set to map the overall composition presented by the LPCEL. An example of the shared attributes for all the visual elements is depicted in Fig. 3.

Each group of the visual elements has its own structure and required fields, although some elements have their own set of attributes, for example, the (12) Data Object has attributes to store local resources or web services.

3.3. Learning patterns

When a good solution to a recurrent design problem is found, then it can be used as a pattern. Patterns must be abstract enough to avoid implementation details and must provide sufficient information about their potential uses. Patterns must be formally documented, and correctly applied to create a feasible solution to a problem (McAndrew et al., 2006).

The use of patterns for educational purposes has been widely studied and some tools have proposed basic approaches to manage patterns (e.g. LAMS editor allows the creation of files that contain design solutions). However, there is a need for a formal specification to store, use and manage patterns in educational environments, which may lead to one of the

```
{
   "id": 1,
   "elementType": "basicElement",
   "complexType": "basic",
   "name": "New element",
   "description": "A new element",
   "educationalAims": [],
   "position": {
      "x": 145,
"y": 50,
      "width": 50,
"height": 50
   },
   "incomingConnections": [],
  "outgoingConnections": []
}
```

Figure 3 Example of a basic activity in the intermediate representation.

main advantages of using patterns: the non-expert users can learn from expert instructional designers or highly skilled teachers.

In the software engineering field, patterns have been divided into different categories (e.g. creational, structural, and behavioral) according to the need the pattern is intended to solve. However, for educational purposes, we propose only two different levels to help users to have a better control over the patterns they create: *High-level detail* and *Low-level detail*.

The main difference between these groups is the level of detail they offer. The high-level patterns provide the overall set of activities that are yet to be fully described by the user, generally, this kind of pattern includes the complex elements depicted in Fig. 2. The low-level patterns are the more detailed sequences of activities, which are represented by the basic elements with already defined sequences and resources. For example, a high-level pattern could be the POL (Project-Oriented Learning) method applied for a midterm assessment, while the low-level detail could be the sequences of activities for each element in that method.

Based on the classification proposed for the software engineering patterns (Gamma et al., 1993), we suggest a structure for describing learning patterns. The objective of the structure depicted in Table 1, is to provide the required formalization in order to achieve a good administration and use of patterns.

3.4. Collaborative design of learning scenarios

The LPCEL Editor provides the mechanism for the users to participate actively in the collaborative design of learning scenarios. The collaboration scheme in the LPCEL is divided into three levels.

3.4.1. Level 1

Introduces the basic mechanisms for the collaboration process where the users are able to change every element of the scenario without restrictions. Each modification is updated for all of the connected users. In this level, the version control is in charge of: (1) the *conflict management* for situations in which two or more users modify the same element at the same time and (2) the *base line supervision* which submits a properly merged version of the scenario to the *repository*.

This level is rather chaotic for learning design because there is no control on the changes that are done to the scenario; everyone can add, modify and delete elements at will. However, this basic scheme is necessary to support the following and more complex levels of collaboration.

3.4.2. Level 2

Includes level 1 and introduces the *per-element restrictions*, which are set during the lifetime of the design process. The owner of the learning scenario defines the elements that are going to be submitted to an agreement when modified, e.g. the owners decide that all users must agree on all the modifications on parallel-type elements. This could be useful to avoid changes on critical parts of the scenario.

Users have their own *local version* to which they can submit changes. This local version is an interesting component because it can *partially* merge with the baseline on the elements or tasks that do not have restrictions. The local version can only be *fully* merged with the baseline until other users approve all the changes.

The users invoke the *agreement system*, depicted in Fig. 4 when they need to authorize some changes in the learning scenario, i.e. it requires the approval of all the users before submitting a new baseline. The objective is to have a better control over the stable version of the scenario. When the agreement system is activated, the involved users review the current version and agree on which changes are applied or discarded. When the agreement is over, the approved version is sent to the control system, which merges the baseline and the new version. Finally, the new version is updated to all users.

3.4.3. Level 3

This level includes Level 1 and 2 and introduces *role-based restrictions* to improve consistency of versions and to avoid modification from non-authorized users. These restrictions allow users to modify only some elements, or parts of the scenario that are linked to their profile, for example, a web designer may only change the *Resources* of the scenario (web pages containing a topic explanation) but cannot modify the sequence of activities related to those resources.

Users are able to choose from the basic configurations of roles and privileges, or to create a configuration of their own. The relationship between roles and rules a reset by element type or actions that can be done with these elements.

Those three levels incorporated in the editor allow users to participate in a collaborative authoring process. The collaboration is highly customizable to meet user needs. The LPCEL Editor also serves as a repository of learning scenarios, where

Table 1 Learning pattern example.

Field	Example
Name	Weekly evaluation
Intent	Create a simple sequence of tasks to implement a weekly review of topics
Alias	Quick quiz, quiz, weekly review
Motivation	This pattern can be used when there is a need to evaluate the progress of a student once a week
Applicability	If there is a need to create a simple sequence of activities related to the weekly roundup, use this pattern
Structure	Visual representation of the pattern
Participants	Start, end and intermediate activities
Contributors	This pattern is executed exclusively in a sequential order
Consequences	Measures the students' level of comprehension, but is not guaranteed that they will perform the readings in depth
Implementation	Is recommended to make other learning techniques in case the students do not like reading
Code	Intermediate representation using the JSON structure
Applications	The timing can be adjusted to apply monthly or final assessments
See other	Monthly evaluation, final evaluation, quick assessment

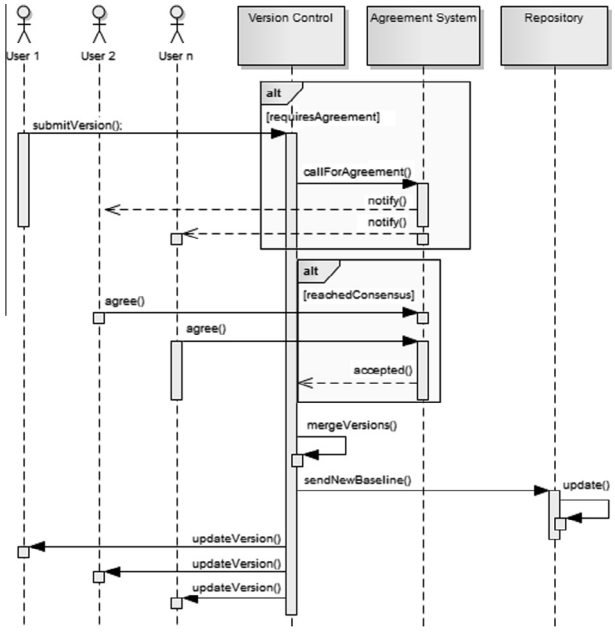

Figure 4 Agreement protocol.

users can store several versions of the same scenario in order to roll back to a version when required.

3.5. Use of web services

A web service is a software system designed to support interoperable machine-to-machine interaction over a network and is normally described using the WSDL standard (Vogten et al., 2006).

The LPCEL Editor supports the integration and testing of web services. It provides the tools needed to include and test educational web services for learning scenarios. Basically, the user only makes use of a wizard where some of the key components of the web services are requested (the URL, port, service, operation, etc.) while the remaining information is gathered automatically. Before the process is completed, the editor provides the mechanism to test the web service so the user can be sure that the selected web service is still functional. The objective of testing a web service during the design stage is to verify if a particular web service or operation is still available, this way the user can check if the service is still operational in order to avoid runtime errors in the execution stage of the learning scenario.

4. A case study of using the LPCEL Editor

The LPCEL Editor has been subject to a case study evaluation (Hevner et al., 2004) in a course that follows a Project-Oriented Learning method (Martín, 2005). In the Monterrey Institute of Technology, Mexico, inside the B.S. Computer Systems Engineering, there is a modality in which some courses are gathered together so the students are able to deliver a complex project. The objective is to simulate real life scenarios in which they will make decisions in order to deliver what the clients demand.

During the course, the students must deliver an incremental version of the system that must be assessed; each milestone consists of the following tasks: the leader manages the requirements and assigns them to the other teammates; the programmer implements the requirements as planned; the tester verifies the work done by the programmer; and the documenting one writes the software documentation. Once these stages are done, it is evaluated first by the client and later by teachers, who assess students according to the specific criteria of an evaluation instrument (i.e. a rubric). Finally the teachers decide the final grade for each one of the students in their respective course. The following steps describe the method of authoring a CLP that uses an external web service to assess the students' work.

4.1. Visually editing the learning process

If users designing this course had to choose a specific EML, it would be rather complicated to use the low-level XML details of the chosen specification. However, this task can be facilitated by using the visual elements provided by the LPCEL Editor. For example, Fig. 5 presents only a small part of the XML low-level details of the LPCEL representing the activities of the case study; it focuses on the set of roles and the first two activities for the students (Manage Requirements and Implement requirements), the definition of the roles (leader, programmer, tester and documenter), as well as relationship between the task and the student who will perform it.

Creating the XML representation could be a difficult task for non-expert users, since it requires a deep understanding of the LPCEL specification in terms of the XML tags and structure, and how to link each component, i.e. which student must perform which task.

Using the visual elements provided by the LPCEL Editor can facilitate the authoring task. The same course design and

```
<LPCELxsi:noNamespaceSchemaLocation="lpcel.xsd">...
<!-- Define the roles for the students-->
<Roles><Roleid="ROLE_1"><Title>Student</Title>
<RoleHierarchy>
<Roleid="ROLE_2"><Title>Leader</Title></Role>
<Roleid="ROLE_3"><Title>Programmer</Title></Role>
<Roleid="ROLE_4"><Title>Tester</Title></Role>
<Roleid="ROLE_5"><Title>Documenter</Title></Role>
</RoleHierarchy></Role></Roles>...
<!-- Define the activities -->
<Complex-Learning-Process><Component-CLPid="ID_1">
<Basic-Component><Action>
<Component-Activityid="ID_2"><Context-Activity>
<Title>Manage Requirements</Title>
</Context-Activity></Component-Activity>
</Action></Basic-Component>
<Basic-Component><Action>
<Component-Activityid="ID_3"><Context-Activity>
<Title>Implement requirements</Title>
</Context-Activity></Component-Activity>
</Action></Basic-Component>...
</Component-CLP></Complex-Learning-Process>  ... <!--more activities->
<!-- Define the activity-role relationship -->
<Assignments>
<AssignmentexecuteOnComponentActivity="ID_2"
            roleAssignment="ROL_2"/>
<AssignmentexecuteOnComponentActivity="ID_3"
            roleAssignment="ROL_3"/>...
</Assignments>...
</LPCEL>
```

Figure 5 Part of an XML representation of the case study.

sequence of activities mentioned in the case study can be represented using the LPCEL Editor visual approach as depicted in Fig. 6. One of the main properties of the visual representation is how easy it presents the pedagogical intention of the sequence of tasks, and how intuitive and understandable it can be for the teacher. It is important to notice how the learning scenario structure (sequence of activities, the roles and which tasks every role performs) can be understood just by analyzing the visual representation.

The intermediate representation, which is based on JSON, can be read and understood by the user. This human-readable property becomes important when the user only needs to do minor adjustments to the course design. Using the intermediate representation is possible to generate courses compliant with a specific EML. The transformations are yet to be implemented in the LPCEL editor, as they have been tested in the model-driven LDDSL approach (Dodero et al., 2012). Although that is out of the scope of this work, it must be noted that a one-to-one mapping between two different EMLs is not always possible without losing expressiveness.

4.2. Use of learning patterns

The LPCEL Editor proposes a structure based on two levels of classification for patterns: The high-level and low-level detail. To illustrate the different levels, we make use of the same case study. The modality presented in the case study is based on the Project-Oriented Learning (POL) method. The POL method consists of a set of high-level sequence of activities and roles (Martín, 2005). This high-level detail of activities can be seen in Fig. 7, and it presents a set of complex activities such as the parallel, sequence, and sub-process. Notice that this is only a part of the complete POL process, but is a good example of the nature of the high-level classification for patterns; it only presents some high-level activities but not the specific tasks.

In the same scenario, if the users needed a more specific arrangement of tasks for the sequence activity, the low-level classification should be used. For example, a low-level description for the individual activity could be a sequence of tasks that can be executed depending on the result of a test (Activity A). The low-level detail category stores complex sequences that have specific names, resources, structures and flows, yet such a

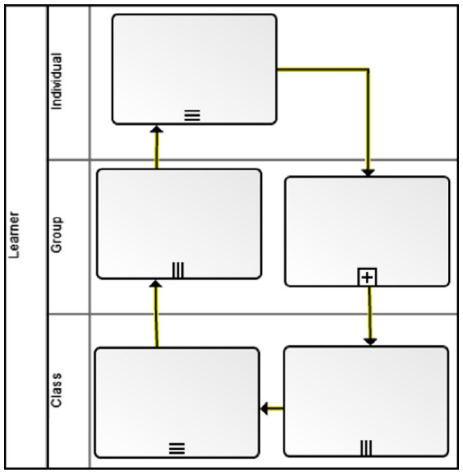

Figure 7 Example of a POL high-level detail pattern.

structure can be easily adjusted to meet the needs found in other design scenarios if needed.

If the complex sequence described above is a good solution for a recurrent problem, then the user can set the complete structure as a low-level pattern as depicted in Fig. 8. It must be saved as low-level detail pattern since this complex sequence contains specific names, flows and resources.

4.3. Collaborating authoring

The level of complexity in the course requires the knowledge of several people to properly design this scenario, e.g. several instructional designers, graphic designers and teachers. All of them would have specific tasks:

- *Instructional designers:* They design the structure of the course, such as the learning technique, the sequence of activities, the roles and other elements. They adjust the general learning process and apply the best practices that guarantee a quality course.
- *Teachers:* This group of people designs the overall content of the activities (learning objectives, learning content, exercises, etc.) in terms of the required resources that each activity needs.

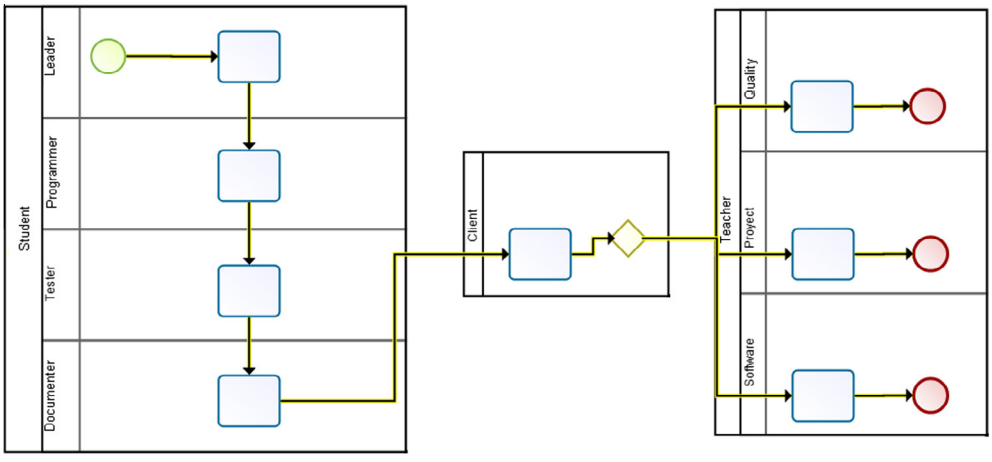

Figure 6 Visual representation of the case study.

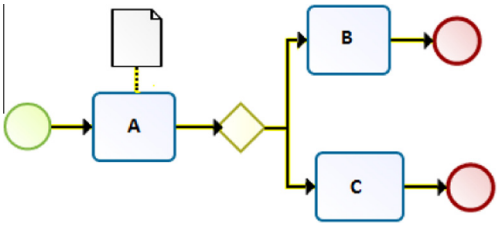

Figure 8 Low-level detail pattern example.

- *Graphical designers:* These people are responsible for designing all the educational content, e.g. the presentations, animations, videos and the appearance of the content generated by the teachers.

In level 1, all users are able to modify the structure of the scenario without any restrictions. Fig. 9 presents an example in which different users (1–3) move some elements of the scenario, and then those changes are updated automatically to all users, generating a new version (baseline). Although this is a basic and uncontrolled behavior, it is necessary to support the advanced levels.

Level 2 allows the owner of the course to set restrictions over some elements and the types of changes that require an agreement to be accepted. For example, assume the owner has set restrictions on any modification done in the parallel and sequence elements, meaning that any changes done in those elements requires an agreement and that any other changes would be automatically updated to the users (as in level 1).

Fig. 10 presents an example of (a) a scenario that is being modified, (b) changes done by a particular user (green lines represent new elements and red lines represent deleted elements) and (c) the result of the scenario following the criteria set by in level 2. The result shows how the elements with no restrictions (the basic elements) are automatically updated, while the changes done in the parallel and sequence elements are not yet updated since they require the agreement of all users. In this case, the changes done in those elements are updated until the agreement process is done.

Agreements can also occur at the level of actions performed on the elements, for example, when renaming the element, changing some of its properties or changing its predecessors or ancestors. This ensures that elements considered as important cannot be modified without restraint.

The level 3 is similar to the level 2, but it adds restrictions based on users' roles, for instance, setting the proper privileges to represent the role descriptions mentioned earlier. The restrictions provide a better control over the generated versions of the learning scenarios because they avoid changes in the elements by non-authorized users, e.g. it has no sense to allow graphical designers to modify the sequence of activities if they are not experts in the subject.

Roles are also used when a deadlock occurs during the agreement process, for example, if there is no consensus about the changes done in the scenario, then the user with the highest level of authority can accept or discard changes at will.

4.4. Web services

In Fig. 8 an external resource was specified for the *Activity A* using the *resource element*. However, the LPCEL specification requires some information about the web service that is being set for the activity. Using the XML tags, the process of assigning web services to activities can be very difficult for two reasons. One is that there is some information that the user needs to get from the WSDL file describing the web service, and a non-expert user might not know where to get such information. Two, the user must place this information in the right structure using the appropriate LPCEL XML tags. Of course, the same task is just as difficult when it comes to writing the same web service description for other EMLs (if the specification supports this characteristic).

Figure 9 Collaboration in level 1.

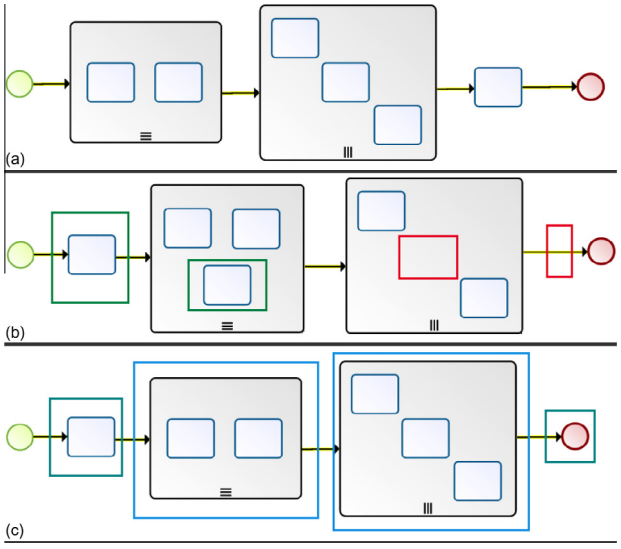

Figure 10 Collaboration in level 2.

Fig. 11 presents only the critical information (URL, Service name, port, namespace etc.) that has to be set to describe a web service in LPCEL.

The JSON structure is used as an alternative to the regular XML-based representation. It is important to highlight how easy it is for the user to attach an external web service to an activity using the LPCEL Editor. In this case, such a process can be accomplished using the Web Services Wizard as presented in Fig. 12.

In the wizard, the user selects the web service and the desired operation, configures the web service client (web forms) and optionally is able to test the web service. All the work needed to complete the XML description of the web service can be summarized in three simple steps where the user only selects the same basic information, but in an intuitive and easy-to-use manner. After completing the steps of the wizard, the LPCEL editor gathers automatically all the relevant information about the web service and a web client is automatically generated.

Using the wizard, users set all of the information about the web services that will be used by the activities in the execution stage. There is no need for the user to write XML tags to describe the service. Also, the user can verify the correct functionality of the web service or even if it is still functional.

```
{
    "id": 1,
    "elementType": "dataElement",
    "complexType": "basic",
    "name": "Web Service Test",
    "description": "Testing",
    "position": {
       "x": 223,      "y": 280,
       "width": 50,  "height": 60
},
    "resource": {
       "type": "remote",
       "url": "http://footballpool.dataaccess.eu/data/info.wso?wsdl",
       "port": "InfoSoap",
       "operation": "StadiumInfo"
    }
  }
```

Figure 11 Intermediate representation of a web service.

5. Survey on user experience

The survey of user experience with the tool has been proposed to corroborate if the tool not only meets the functional requirements, but also serves to facilitate the design of scenarios from the view point of those who would use it (teachers, instructional designers, graphic designers, etc.).

The survey is focused on three main elements of the editor: (1) Visual elements, (2) Patterns and (3) Collaboration. The survey was answered by 55 people who are experts on the design of learning scenarios. Although the population is rather small, their level of experience and knowledge represent a good opportunity to determine which aspects of the editor are most relevant in terms of instructional design.

5.1. Visual elements

The objective is to measure the user's perception in terms of how they interact with the visual elements and their potential to represent learning scenarios. The scale goes from 1 to 5, where the first is the lowest grade possible, and the later means users are completely satisfied or agree. The results are presented in Fig. 13.

The first and second questions (1. How intuitive are the visual elements? 2. Do you think some training is needed to learn the proper use of the visual elements?) show that while the elements are fairly intuitive, training is still required to make a good use of the elements. The third and fourth questions (3. How simple was it to represent a learning scenario? 4. Are the visual elements enough to represent them?) show the level of simplicity provided by the visual elements to represent scenarios; is evident that some level of knowledge about the elements is required and there's room for improvement in terms of the iconography used. Finally, the fifth and sixth questions (5. Is it simple to understand the learning scenario using the visual elements? 6. In brief, how useful it is to use the visual elements to represent learning scenarios in comparison to traditional methods?) provide an insight about the overall simplicity of the visual elements to describe learning scenarios in comparison to the traditional methods they use, emphasizing how simple it is for the user to understand the scenarios presented in the editor.

5.2. Learning patterns

This section of the survey is focused on the use of patterns for the creation of learning scenarios and the need of users for having tools to manage patterns in a simple fashion. The scale is yes or no for question one; and goes from 1 to 5 for question number two and three, where the first is the lowest grade possible, and the later means users are completely satisfied or agree.

The first part of this section, presented in Fig. 14, inquires into the need of users to use best practices to solve recurrent problems. The answers to questions (1) Do you think it is important to create a course using best practices? (2) To what extent do you think it facilitates the creation of full courses? and (3) How useful it would be to have a tool which manages

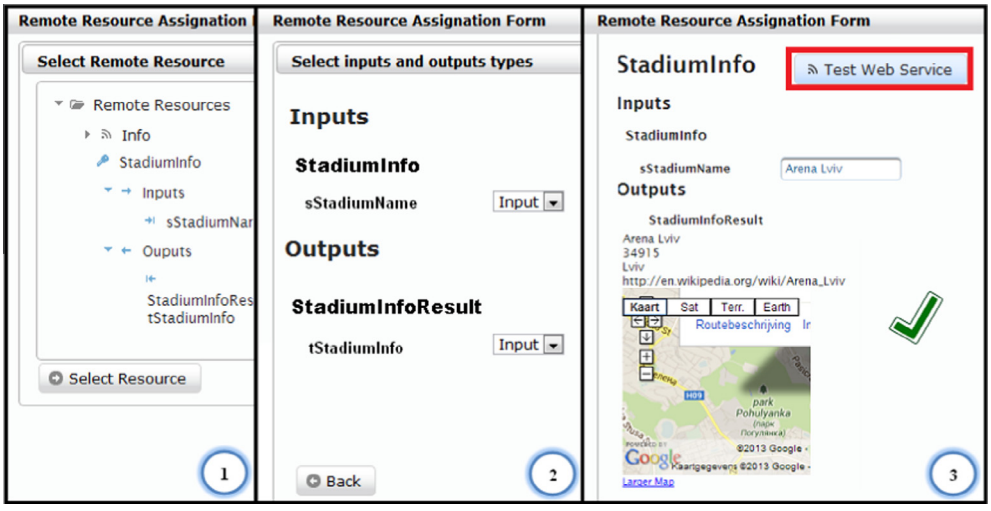

Figure 12 Web service wizard.

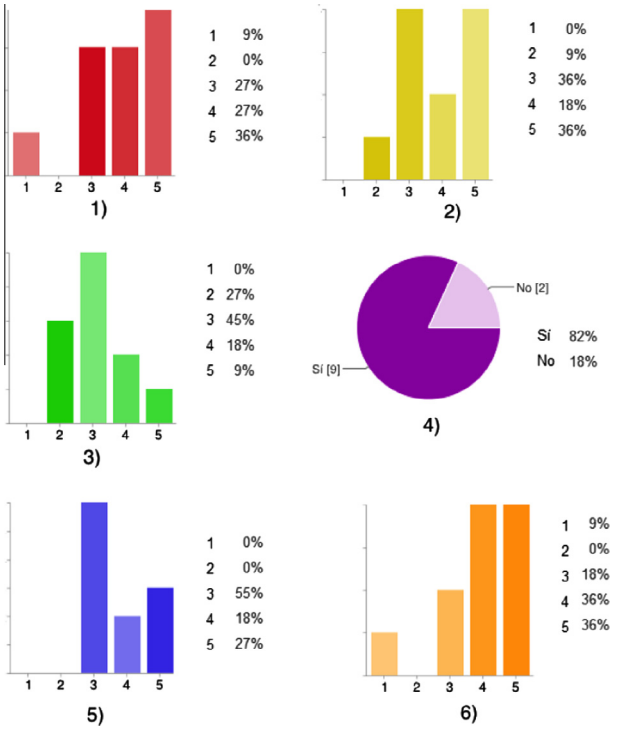

Figure 13 Section for visual elements.

educational patterns? These questions show that course creators use patterns in regular basis to build learning scenarios and they need a tool to assist this process.

The second part of this section explores the elements that should be kept and improved into the pattern manager tool from the point of view of users. In Fig. 15, we present some of the highest rated components of the tool that users categorized as *very important* or *mandatory*. Users evaluated various components that were included in the pattern management tool, however, we only show those components that were well evaluated by the users in terms of usability, user experience and usefulness. The specific components are: (1) Pattern Classification, (2) Pattern search engine, (3) Formal structure and (4) Pattern personalization and usage. Users commented that these components would be helpful for most of the tasks related with the usage of patterns.

Given the need of users for a mechanism to manage learning patterns and the capabilities of the tool, we consider that this part of the editor was well received, since it covers the basic functionality demanded by the users, while it makes the task of reusing patterns very simple.

5.3. Collaborative learning design

This collaboration among users during the design of learning scenarios is one of the main components of the editor. In this part of the survey, users are asked about how they collaborate and their experience with the editor in this regard.

Fig. 16 presents the results for questions (1) How often do you require from collaborators to design learning activities?

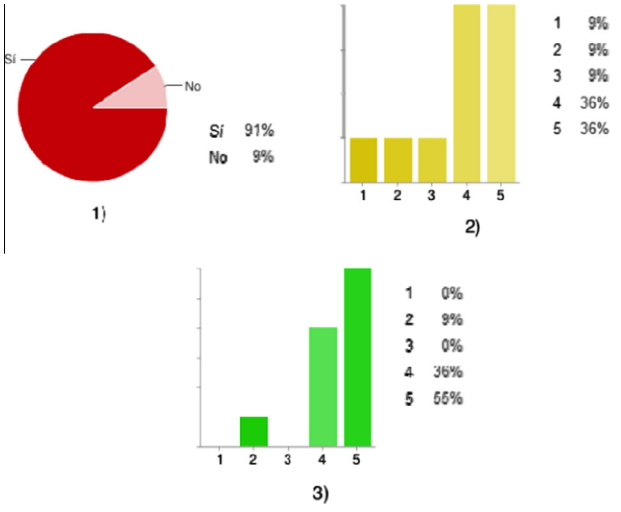

Figure 14 Section for learning patterns.

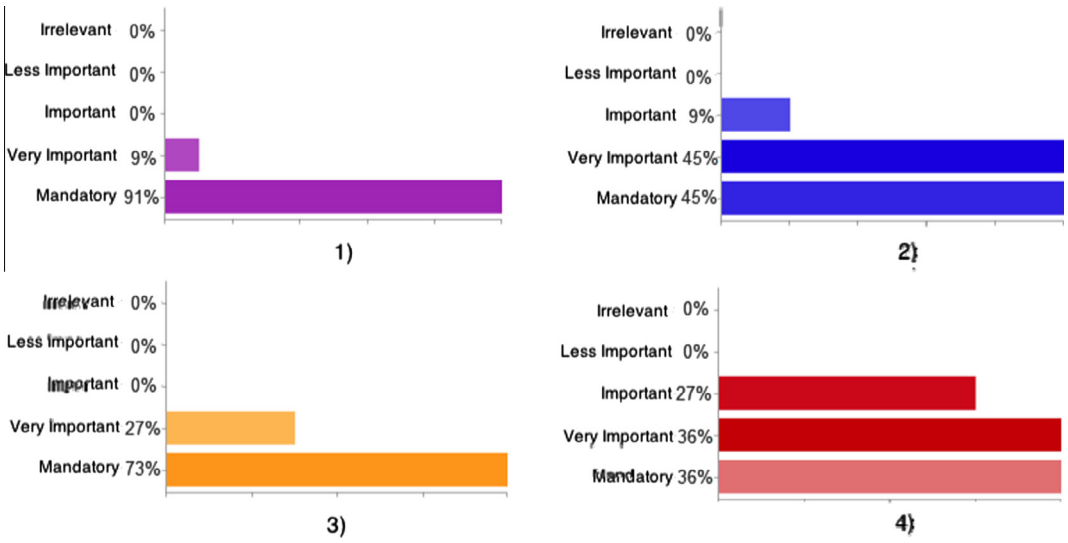

Figure 15 Highest rated components for the pattern manager.

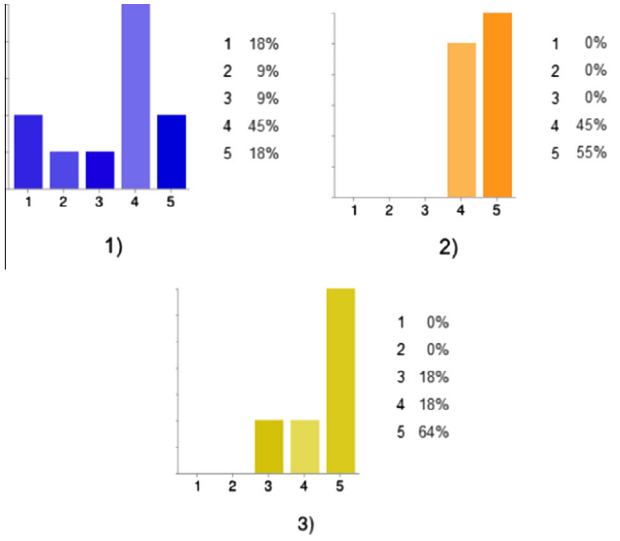

Figure 16 Section for collaborative learning design.

and (2) How necessary it is to design a course between several people with different profiles? and (3) How useful it is to design a course using the tool for the collaboration process? The scale is the same as in previous sections, from 1 being the lowest grade, to 5 meaning that users agree.

Question number one and two are interesting since both show that users are collaborating, but the methods they use are rather complicated and ineffective, e.g. they design the scenarios using text files and each change is notified via e-mail, resulting in conflicts. These results were a clear indication that an automated tool for real-time collaboration is needed among users designing learning scenarios.

Given these results, users worked in a collaborative way in the design of a complex learning scenario using all the other mentioned tools and the components for collaboration. Question number 3, represents the summary of the user experience and feedback about the collaboration tool, in which it is clear that most users found the tool easy to use and better for real-time collaboration purposes than their traditional methods. An important aspect of highlighting is that most users were able to make a good use of the tool with little to no explanation.

6. Conclusions

In this work, we have developed a visual authoring tool compliant with the LPCEL specification. The objective of creating the tool is to ease the authoring process of learning scenarios. It was a key reason for us to choose the LPCEL as the target EML because its level of expressiveness is broader than IMS LD or LAMS LD.

Using a visual approach for the authoring process allows the user to generate a learning scenario in a simple way. This process can also be facilitated through collaboration among several users. In most cases, several people create a learning scenario, each one focusing on different parts of the scenario. Therefore, authoring tools should also aim for a collaborative authoring process capable of handling the user requirements.

User experience with the tool, suggests that the features included are necessary to perform an instructional design simpler and more closely to the requirements of users. Although a deeper analysis is still needed, these preliminary results indicate that the editor can serve its purpose.

There are some characteristics that need to be improved, such as the EML-specific converter which are yet to be implemented and because a 1–1 transformation is not possible, there are some issues that must be handled accordingly in such a process. There is also the need for an LPCEL player within the editor so the user can test or simulate the learning scenario.

References

Atkinson, C., Kuhne, T., 2003. Model-driven development: a metamodeling foundation. Software, IEEE 20 (5), 36–41.

Caeiro Rodríguez, M., Derntl, M., Botturi, L., 2010. Visual instructional design languages. Journal of Visual Languages & Computing 21 (6), 311–312.

Crockford, D., 2006. The Application/Json Media Type for JavaScript Object Notation (JSON). The Internet Engineering Task Force.

Dalziel, J.R., 2003. Implementing learning design: the learning activity management system (LAMS). In: Crisp, G., Thiele, D., Scholten, I., Barker, S., Baron, J. (Eds.), Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE), pp. 593–596..

De la Fuente, L., Miao, Y., Pardo, A., 2008. A supporting architecture for generic service integration in IMS learning design. In: European Conference on Technology-Enhanced Learning **5192**, Springer Verlag, pp. 467–473.

Dodero, J.M., Martínez del Val, Á, Torres, J., 2010. An extensible approach to visually editing adaptive learning activities and designs based on services. Journal of Visual Languages & Computing 21 (6), 332–346.

Dodero, J.M., Ruiz-Rube, I., Palomo-Duarte, M., Cabot, J., 2012. Model-driven learning design. International Journal of Research and Practice in Information Technology 44 (3), 61–82.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1993. Design patterns: abstraction and reuse of object-oriented design. In: Proc. of ECOOP, Springer Verlag, pp. 406–431.

Griffiths, D., Beauvoir, P., Liber, O., Barrett-Baxendale, M., 2009. From reload to ReCourse: learning from IMS learning design implementations. Distance Education 30 (2), 201–222.

Hernández-Leo, D., Pérez, J., Dimitriadis, Y., 2004. IMS learning design support for the formalization of collaborative learning patterns. In: Proc. of the fourth International Conference on Advanced Learning Technologies IEEE, IEEE Computer Society, pp. 350–354.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Quarterly 28 (1), 75–105.

Koper, R., 2001. Modeling Units of Study from a Pedagogical Perspective: The Pedagogical Metamodel Behind EML. Open University of the Netherlands. Educational Technology Expertise Center, Heerleen.

Koper, R., Tattersall, C., 2005. The learning design specification. In: A Handbook on Modelling and Delivering Networked Education and Training. Springer, pp. 21–40.

Laforcade, P., 2007. Visualization of learning scenarios with UML4LD. Journal of Learning Design 2 (2), 31–42.

Martín, M., ITESM (Eds.), 2005. El modelo educativo del Tecnológico de Monterrey. Tecnológico de Monterrey.

Martínez-Ortiz, I., Moreno-Ger, P., Sierra-Rodríguez, J., Fernández-Manjón, B., 2008. A flow-oriented visual language for learning designs. Advances in Web Based Learning (ICWL) 1, 486–496.

Martínez-Ortiz, I., Sierra, J.L., Fernández-Manjón, B., 2009. Authoring and reengineering of IMS learning design units of learning. IEEE Transactions on Learning Technologies 2 (3), 189–202.

McAndrew, P., Goodyear, P., Dalziel, J., 2006. Patterns, designs and activities: unifying descriptions of learning structures. International Journal of Learning Technology 2 (2), 216–242.

Murta, L., Corrκa, C., Prudκncio, J., Werner, C., 2008. Towards odyssey-VCS 2: improvements over a UML-based version control system. In: Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models, ACM New York, NY, USA, pp. 25–30.

Padrón, C., Dodero, J., Lanchas, J., 2005. CASLO: collaborative annotation service for learning objects. IEEE TCLT Learning Technology Newsletter 1 (ISSN 1438-0625).

Paquette, G., Marino, O., De la Teja, I., Lonard, M., Lundgren-Cayrol, K., Contamines, J., 2005. Implementation and deployment of the IMS learning design specification. Canadian Journal of Learning and Technology 31 (2).

Paquette, G., Léonard, M., Lundgren-Cayrol, K., Mihaila, S., Gareau, D., et al, 2006. Learning design based on graphical knowledge-modeling. Journal of Educational Technology and Society Special issue on Learning Design 9 (1), 97–112.

Sampson, D., Karampiperis, P., Zervas, P., 2005. ASK-LDT: a web-based learning scenarios authoring environment based on IMS learning design. International Journal on Advanced Technology for Learning (ATL) 2 (4), 207–215.

Torres, J., Dodero, J., Aedo, I., Diaz, P., 2006. Designing the execution of learning activities in complex learning processes using LPCEL. In: Advanced Learning Technologies, 2006. Sixth International, IEEE Computer Society, pp. 415–419.

Torres, J., Juarez, E., Dodero, J., Aedo, I., 2009a. EML learning flow expressiveness evaluation. In: Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International, IEEE Computer Society, pp. 298–300.

Torres, J., Reséndiz, J., Dodero, J.M., Aedo, I., 2012. LPCEL Editor: a web-based visual authoring tool for learning design. In: Proc. of the 12th IEEE International Conference on Advanced Learning Technologies (ICALT 2012). IEEE Computer Society, Rome, Italy, pp. 141–142.

Vogten, H., Martens, H., Nadolski, R., Tattersall, C., van Rosmalen, P., Koper, R., 2006. CopperCore service integration – integrating IMS learning design and IMS question and test interoperability. In: 6th International Conference on Advanced Learning Technologies, IEEE Computer Society, pp. 378–382.