International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(B)**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **845–865**

INTEROPERABILITY BETWEEN VISUAL UML DESIGN APPLICATIONS AND AUTHORING TOOLS FOR LEARNING DESIGN

Roberto Barchino*, José R. Hilera, Luis De-Marcos, José M. Gutiérrez
Salvador Otón, José A. Gutiérrez, José J. Martínez and Lourdes Jiménez

Computer Science Department
University of Alcalá
28871 Alcalá de Henares, Spain
*Corresponding author: roberto.barchino@uah.es
{ jose.hilera; luis.demarcos; josem.gutierrez; salvador.oton }@uah.es
{ jantonio.gutierrez; josej.martinez; lou.jimenez }@uah.es

Received October 2010; revised February 2011

Abstract. *A learning design is the result of the description of a teaching-learning process. It implies the consideration of aspects such as the context in which teaching is being developed, the most suitable teaching method in this context, the necessary resources, the contents of the training activity, or the assessment criteria. IMS-LD is a specification which provides a containment framework of elements that formally describe any teaching-learning process design, and for this reason it supports the description of learning designs using specific authoring tools. It is common to create Activity Diagrams, using the Unified Modelling Language (UML) notation, to represent with a model the sequence of learning activities involved in a learning design and it is done using a visual UML design application. In this paper, a method and software application (based on XMI and XSLT technologies) that ensure interoperability among UML modelling tools and IMS-LD editors are presented.*

Keywords: UML, Learning design, XML

1. Introduction.

The term "learning design" is often used to describe the outcomes of the process of designing, planning and orchestrating learning activities. Different modelling languages based on XML have been created to represent learning designs. One of the best known is IMS-LD [1], a pedagogically neutral specification based on EML (Educational Modelling Language) [2], the goal of which is to provide a containment framework of elements that can formally describe any teaching-learning process design. IMS-LD defines labels to represent the different elements specified when designing a teaching-learning process, such as teaching objectives, participants (roles), activities to perform, the teaching method applied and learning objects used.

The main problem of these modeled languages is that actually they are very little used, because normally teachers who would have to use them for the educative activities design are not experts in the application of formal languages, which are more known in the scope of the computer science than in the one of the education. At the moment, there is a research field in e-learning called "Visual Instructional Design Languages" (VIDL) [3] which tries to offer the teachers more abstract languages, of visual or graphical type, and whose use is more intuitive.

Some of the visual languages that have been proposed recently are Flexo [4], E2ML, PoML or CoUML [5]. Generally, they have to offer the possibility of realizing several types of diagrams and sometimes, some of them are used to describe graphically the order or the sequence in which activities of a learning process must be executed, drawing a workflow diagram. Several different modelling notations and languages can be used to represent sequences of activities in a workflow. At the present, it is possible to differentiate between two different model-driven approaches to modeling learning processes [6]: an approach based on adapting generic modeling languages, such as BPEL (Business Process Execution Language) or the UML (Unified Modeling Language) [7] standard[1], and a second approach based on creating domain specific modeling languages. In this latter case, such languages are oriented towards modelling e-learning processes.

Although several different approaches exist in the IMS-LD original specification document, the diagramming technique used to describe examples of teaching-learning processes in this document by IMS is the Activity Diagram, which is part of UML. It enables workflows to be represented as a set of activities connected through transitions subject to certain conditions (for example, when all students in a group have to complete the exercises before moving on to the next stage of a course or other more complex as those related with uncertainty [10]). In this type of diagrams, the actors who must perform each of the activities can also be shown. Most of the UML design applications allow UML diagrams, including activity diagrams, to be exported in a standard file format known as XMI (XML Metadata Inter-change) [11], which ensures interoperability among different modelling tools. XMI is a language to representation and to exchange models and meta-models.

Both IMS-LD and XMI are XML languages, and therefore, it is possible to apply XSLT (Extensible Stylesheet Language Transformations) technology [12], created by the World Wide Web Consortium (W3C) to transform XML documents. This article reports the use of XSLT to assure the interoperability among UML and IMS-LD tools. The motivation of this study is due to the fact that although in the last years different Visual Instructional Design Languages have appeared, it can be demonstrated that in most of them it is included a type of diagram to realize the flow charts or activities diagrams, and in many cases, they are based on UML. According to the authors of this article, it is more efficient to realize modelled workflows using an accepted notation like UML and XMI, instead of creating new languages for it. Recently, it has been published a comparative study of different modelled languages that include flow charts, and indeed the language based on activity diagrams of UML was the one which obtained the best results in semiotic clarity, visual expressiveness, semantic transparency and perceived usefulness [5].

The advantages of the solution that appears in this article are: first, it is based on widely accepted standards by the community of e-learning (IMS-LD) and by the community of systems modeled (UML, XMI); and secondly, it offers a tool that can be useful to guarantee the interoperability among the available environments in these two scopes, through the automatic and bidirectional conversion of formats.

In this paper, first, the authors have reviewed general aspects of modelling teaching-learning processes using the IMS-LD specification, created to formalize this kind of models. Secondly, the relationship between UML and learning design has been analysed, from UML meta-models or profiles established to model IMS-LD elements, to the use of UML Activity diagrams for specifying the sequence of learning activities. It is given a description of the main characteristics of an open source software program enabling the interoperability among IMS-LD and UML tools. This performs an automatic transformation of UML Activity Diagrams into IMS-LD format, which may later be processed by an editing/authoring tool for learning design. This software application also allows reverse engineering of learning designs, i.e., the automatic generation of an activity diagram from

[1]UML is now an ISO/IEC standard (namely ISO/IEC 19505) [8,9].

existing learning designs included in IM-LD learning units, so that it can be visualized and modified using UML modelling tools. Finally, some conclusions drawn from this study are discussed.

2. Model Driven Approach to Design Teaching-learning Processes.

A process can be defined as a sequence of activities in which different entities (people, machines, etc.) collaborate to achieve an objective. "Workflow" is a concept related to "process" and refers to the total or partial automation of one or more processes. A Workflow Management System is a computerized system to define the processes that constitute a workflow, and to manage the execution of the processes by means of the interpretation of the models.

The tools for process modeling, such as workflow, use their own technology to store the information that includes definitions of processes. The Workflow Management Coalition[2] (WfMC) has set different standards for sharing information among commercial systems of this type.

Different methods and technologies originally applied within the scope of *workflow systems* have also been applied to e-learning in what is called Workflow-based e-learning systems [11]. In general terms, such developments can be part of the trend called the model-driven approach to design learning processes. Its aim is not only to use an abstract modeling language to specify processes, but also to implement mechanisms to automatically generate learning units in different technological environments using the same learning design [14], and to facilitate new pedagogical issues in this area (for example, those related with competitive and cooperative learning [15]).

Different kinds of abstract languages have been used to specify learning designs. They allow to create platform-independent models (PIMs) and to use different implementation languages, so it is possible to translate the same PIM to one or more platform-specific models (PSMs). The UML or UML extensions, such as CoUML [16], can be used to model learning designs, as well as domain specific languages created for this purpose [17], whilst IMS-LD is the principal mechanism used to implement learning units in e-learning platforms. In this paper, we will focus on UML and IMS-LD and we will present a solution for transforming learning designs modeled with UML into learning units encoded according to the IMS-LD format. However, first, in the following subsections, both languages are described.

2.1. IMS learning design.

In 2003, IMS published its Learning Design (LD) specification. This specification was created using the Educational Modelling Language (EML) proposal from the Open University of Netherlands as a departure point. The objective of the IMS-LD is to "provide a containment framework of elements that can describe any design of a teaching-learning process in a formal way" [1]. The primary use of IMS-LD is to model Learning units (UoL) by including an IMS Learning Design in a content package.

A Learning unit is an abstract term used to refer to any delimited piece of education or training, such as a course, a module and a lesson; it represents more than simply a collection of ordered resources to learn. It includes a variety of prescribed activities (problem solving activities, search activities, discussion activities, peer assessment activities, etc.), assessments, services and support facilities provided by teachers, trainers and other staff members. Activities are the core of the 'learning workflow' model for a learning design. The original IMS-LD document states that an UML activity diagram can be used as a convenient starting point for the design of the learning workflow of a Learning unit.

[2]http://www.wfmc.org

An IMS Content Package is a zip file, and its contents are described in an XML document called the 'package manifest', integrated in the file (Figure 1). To create an UoL, IMS Learning Design is integrated within an IMS Content Package by including in the package manifest the learning design element as another kind of organization within the "Organizations" element.

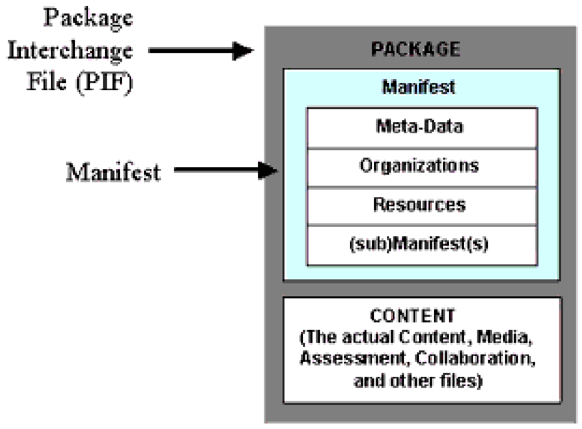

Package Interchange File (PIF)
Manifest
PACKAGE
Manifest
Meta-Data
Organizations
Resources
(sub)Manifest(s)
CONTENT
(The actual Content, Media, Assessment, Collaboration, and other files)

Figure 1. IMS content packaging scope [18]

IMS-LD specifies three levels of implementation and compliance. Learning Design Level A includes everything described so far. Therefore, it contains all the necessary vocabulary to support pedagogical diversity. Levels B and C add three additional concepts and their associated capabilities in order to support more sophisticated behaviors. Learning Design Level B adds Properties and Conditions to level A, enabling personalization and more elaborate sequencing and interactions based on learner portfolios [19]. It can be used to direct the learning activities as well as to record outcomes. The separation of Properties and Conditions into a separate Schema also enables it to be used independently of the rest of the Learning Design Specification, typically as an enhancement to IMS Simple Sequencing. Learning Design Level C adds Notification to level B.

2.2. **UML for learning designs modelling.**

IMS uses UML to provide intuitive graphical models that help to understand the restrictions included in IMS-LD notation. UML is used to model the implementation sequence of activities that have been included in a teaching-learning process; for this, it uses the "Activity Diagram" [7], which permits the representation of workflows as a set of activities connected through transitions under certain conditions.

Soon it will be assessed the usefulness of the UML (Unified Modeling Language) standard in establishing meta-models for the IMS-LD specification and in elaborating dynamic models that reflect the organization of activities in a teaching-learning process.

2.2.1. *UML meta-models and UML profiles.*

The XML tag arrangement necessary to design a learning process is subject to the established rules in the IMS-LD specification, which are given in the aforementioned public schema documents. IMS also offers the UML standard to provide intuitive graphical models that help to understand such restrictions. Such models can be considered actual meta-models, since their objective is not to model the teaching-learning process but to define the mechanisms that designers have at their disposal to create them. Figure 2 shows a simplified representation of the meta-model that corresponds to the core elements defined in the IMS-LD specification.

The IMS-LD conceptual model is expressed as a set of UML class models together with a definition of the vocabulary used, and it represents the overall conceptual model.

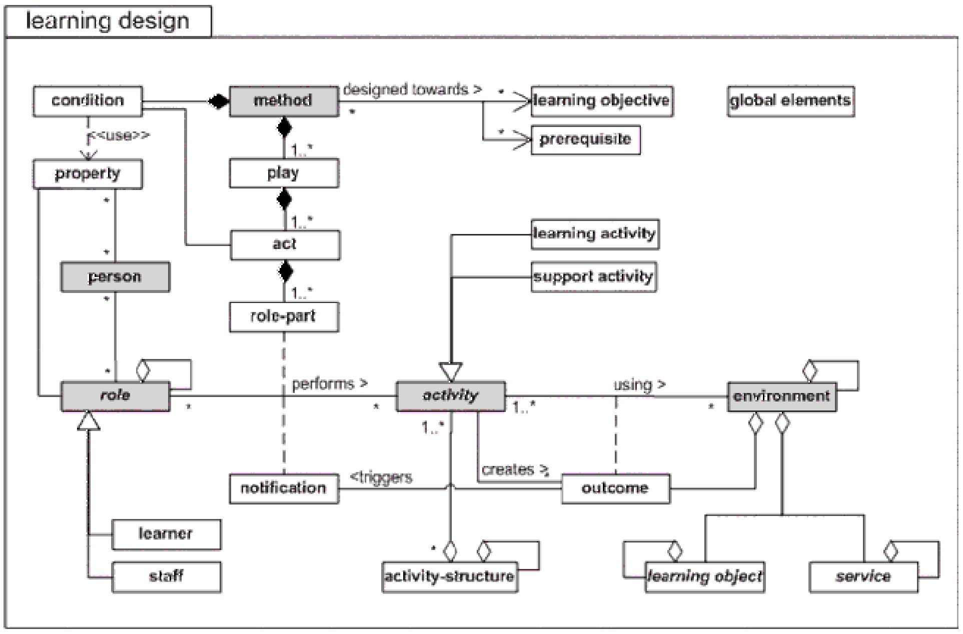


Figure 2. UML meta-model of overall IMS learning design [1]

The core concept of the Learning Design Specification, as it is shown in Figure 2, is that regardless of pedagogical approach, a person is given a role in the teaching-learning process, usually a learner or a staff role. In this role, he or she works towards certain outcomes by performing more or less structured learning and/or support activities within an environment. The environment consists of the appropriate learning objects and services employed during the performance of the activities. To determine which role gets which activities at what moment in the process is done by the method or by a notification.

Other authors have proposed UML profiles to validate learning models. An UML profile provides a generic extension mechanism for customizing UML models for particular domains and platforms. Profiles are defined using stereotypes, tag definitions, and constraints that are applied to specific model elements, such as Classes, Attributes, Operations and Activities. A profile is a collection of such extensions which collectively customize UML for a particular domain (e.g., education, aerospace, healthcare, financial and user interfaces [20]) or platform (Java, .NET).

An example of an IMS-LD profile is the UML4LD [21], which exploit the UML extensions mechanisms in order to make explicit the LD concepts when they are represented with this graphical notation. Another example of a profile is presented in [22] to generate an XML schema that can be used to verify a number of available IMS-LD documents.

2.2.2. *UML activity diagrams for learning designs.* The UML standard can also be used to model the activity execution sequence that has been included in a teaching-learning process. In order to do this, the so-called "Activity Diagram" is used. It enables workflows to be represented as a set of activities connected through transitions that are subject to certain conditions (for example, all students in a group have completed the exercises before moving the next stage of a course). Activity diagrams also show the actors who must complete each action.

An UML activity diagram is a graph that models processes; its primary focus is on the sequence and conditions in which the actions take place. In such a graph, actions represent atomic actions; that is, states that invoke actions and then wait for their completion. Figure 3 shows a sample activity diagram created with an UML tool representing the workflow of activities in an easy learning design, comprising a sequence of two lessons, a selection among two practical work projects, and a final assessment activity.

When the activity model of the learning process has been designed, the next step entails creating the Learning unit that corresponds to this design. A learning design editor, such as Reload LD Editor[3], is usually used. Such an editor enables the learning designer to describe all the elements that comprise the teaching-learning process, including the sequence of activities, to produce a Learning unit (a .zip file) which includes the imsmanifest.xml file with the final description in the XML language defined by IMS-LD.

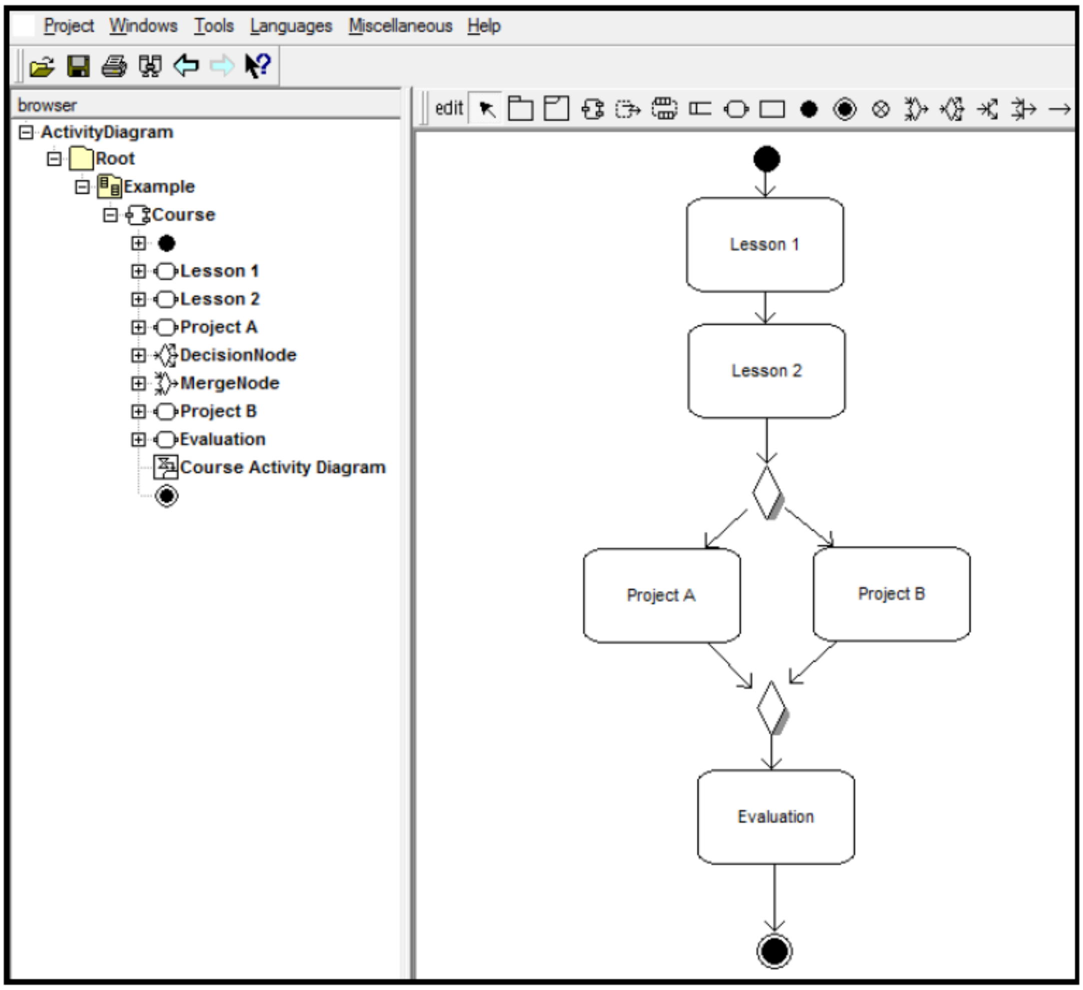


FIGURE 3. Activity diagram of a simple learning design (open source tool used: Bouml[4])

Section 4 presents a new method for automatically producing the manifest for a Learning unit from the activity diagram representing its workflow. In addition, it is also presented a complementary tool that performs the reverse process, namely, that automatically obtains the activity diagram from the learning unit manifest; making easier the re-engineering of the learning processes.

[3]http://www.reload.ac.uk/new/ldeditor.html
[4]http://sourceforge.net/projects/bouml/

3. UML and IMS-LD Tools.

3.1. **UML design applications and XMI format.** An UML tool or UML modelling tool is a software application that supports some or all of the notation and semantics associated with the Unified Modelling Language. On the Web, it is possible to find different comparative analyses of UML tools[5]. Such listings are useful because there are many options and many of them are open source tools. One of the features to consider when choosing an UML tool is the possibility of importing and exporting the models using XMI format. XMI (XML Metadata Interchange) is the format for UML model interchange.

One function of XML Metadata Interchange (XMI) is to enable easy interchange of metadata among UML-based modelling tools and MOF-based metadata repositories in distributed heterogeneous environments [23]. XMI is also used as the way by which models are passed from modelling tools to software generation tools as part of model-driven engineering.

XMI integrates four industry standards: XML (eXtensible Markup Language, a W3C standard), UML (Unified Modelling Language, an OMG modelling standard), MOF (Meta Object Facility, an OMG language used to specify metamodels) and MOF Mapping to XMI. XMI architecture simplifies communication among applications using different technologies, thus it allows saving a great amount time and effort. Furthermore, it also supports the potential reuse of objects and components. To sum up, XMI architecture represents a step towards a complete automation of software development.

Unfortunately, XMI does not yet support graphic diagram interchange. XMI is only able to transport information concerning elements which are contained in an UML model but not the information on how these elements are represented and laid out in diagrams. Thus, if an UML model is stored using an UML tool and then loaded in a different UML tool (or even the same tool) using XMI, all diagram information will be lost. This limitation is not due to XMI itself but rather to the fact that the UML metamodel does not define a standard way of representing diagram definitions.

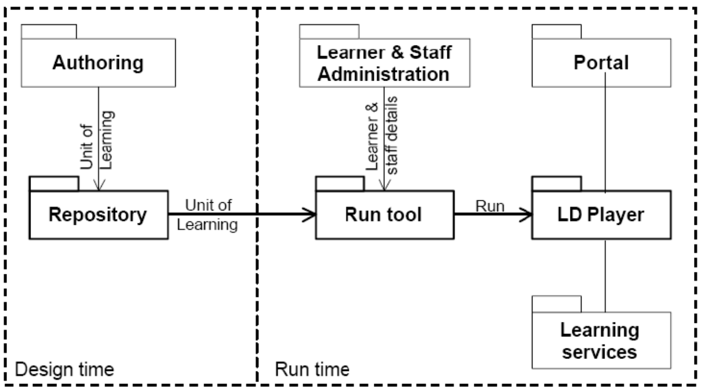


FIGURE 4. Components involved in the design and execution of a IMS-LD learning unit (source: IMS global learning consortium)

[5]Some websites with lists of UML tools are:
http://en.wikipedia.org/wiki/List_of_UML_tools
http://www.dmoz.org/Computers/Programming/Methodologies/Modeling_Languages/Unified_Modeling_Language/Tools//
http://case-tools.org/uml.html
http://www.uml-forum.com/tools.htm

To reduce this problem, the OMG has published UML-DI (UML Diagram Interchange) [24], but many UML tools do not support this extension. UML-DI extends the UML metamodel with a supplementary package for graph-oriented information while leaving the current UML metamodel fully intact. Moreover, it is compliant with the UML 2.0 metamodel and should also be unaffected by any subsequent change or extension.

Within the learning design scope, UML tools can be useful for modeling workflows of teaching-learning actions. UML activity diagrams can be created and then exported to XML files in XMI format. Listing 1 shows a code extract which corresponds to the diagram in Figure 4 (exported as *ActivityDiagram.xmi* file). It contains the XMI labels that represent each diagram element. For example, < *node* > tags are used to represent activities, while < *edge* > tags represent transitions among activities.

```
<?xml version=“1.0” encoding=“UTF-8”?>
<xmi:XMI xmlns:xmi=“http://schema.omg.org/spec/XMI/2.1”
  xmlns:uml=“http://schema.omg.org/spec/UML/2.1.2” xmi:version=“2.1”>
  <uml:Package xmi:id=“Root” name=“Root”>
    <packagedElement xmi:type=“uml:Package” xmi:id=“Example”
      name=“Component View” visibility=“public”>
        <packagedElement xmi:type=“uml:Activity” xmi:id=“Course”
          name=“Course” visibility=“public”>
          <node xmi:type=“uml:InitialNode” xmi:id=“Initial”>
            <outgoing xmi:idref=“Initial-to-Lesson1”/>
          </node>
          <node xmi:type=“uml:CallOperationAction” xmi:id=“Lesson1”
            name=“Lesson 1”>
            <incoming xmi:idref=“Initial-to-Lesson1”/>
            <outgoing xmi:idref=“Lesson1-to-Lesson2”/>
          </node>
          <node xmi:type=“uml:CallOperationAction” xmi:id=“Lesson2”
            name=“Lesson 2”>
            <incoming xmi:idref=“Lesson1-to-Lesson2”/>
            <outgoing xmi:idref=“Lesson2-to-ProjectDecision”/>
          </node>
          <node xmi:type=“uml:CallOperationAction” xmi:id=“ProjectA”
            name=“Project A”>
            <incoming xmi:idref=“ProjectDecision-to-ProjectA”/>
            <outgoing xmi:idref=“ProjectA-to-ProjectMerge”/>
          </node>
          <node xmi:type=“uml:CallOperationAction” xmi:id=“ProjectB”
            name=“Project B”>
            <incoming xmi:idref=“ProjectDecision-to-ProjectB”/>
            <outgoing xmi:idref=“ProjectB-to-ProjectMerge”/>
          </node>
          <node xmi:type=“uml:CallOperationAction” xmi:id=“Evaluation”
            name=“Evaluation”>
            <incoming xmi:idref=“ProjectMerge-to-Evaluation”/>
            <outgoing xmi:idref=“Evaluation-to-Final”/>
          </node>
          <node xmi:type=“uml:ActivityFinalNode” xmi:id=“Final”>
            <incoming xmi:idref=“Evaluation-to-Final”/>
          </node>
          <node xmi:type=“uml:DecisionNode” xmi:id=“ProjectDecision”
            name=“DecisionNode”>
            <incoming xmi:idref=“Lesson2-to-ProjectDecision”/>
            <outgoing xmi:idref=“ProjectDecision-to-ProjectA”/>
            <outgoing xmi:idref=“ProjectDecision-to-ProjectB”/>
          </node>
          <node xmi:type=“uml:MergeNode” xmi:id=“ProjectMerge”
          name=“MergeNode”>
          <incoming xmi:idref=“ProjectA-to-ProjectMerge”/>
          <incoming xmi:idref=“ProjectB-to-ProjectMerge”/>
          <outgoing xmi:idref=“ProjectMerge-to-Evaluation”/>
        </node>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“Initial-to-Lesson1”
          source=“Initial” target=“Lesson1”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“Lesson1-to-Lesson2”
          source=“Lesson1” target=“Lesson2”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“Lesson2-to-
          ProjectDecision”
          source=“Lesson2” target=“ProjectDecision”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“ProjectA-to-ProjectMerge”
          source=“ProjectA” target=“ProjectMerge”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“ProjectDecision-to-
          ProjectA” source=“ProjectDecision” target=“ProjectA”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“ProjectDecision-to-
          ProjectB” source=“ProjectDecision” target=“ProjectB”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“ProjectB-to-ProjectMerge”
          source=“ProjectB” target=“ProjectMerge”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“ProjectMerge-to-
          Evaluation” source=“ProjectMerge” target=“Evaluation”/>
        <edge xmi:type=“uml:ControlFlow” xmi:id=“Evaluation-to-Final”
          source=“Evaluation” target=“Final”/>
      </packagedElement>
    </packagedElement>
  </uml:Package>
</xmi:XMI>
```

Listing 1. XMI code of the activity diagram in Figure 4, exported as Activity-Diagram.xmi file

3.2. **Learning design authoring tools.** At the beginning of the present century, the most commonly used software for supporting teaching and learning interactions in e-learning comprised learning management systems (LMS). These systems provide the means for building courses, managing roles and groups, and building services such as forums, conferencing and chats. However, the course structuring functionality in LMS is typically content-centered; thus the traditional activity management capabilities in LMSs are usually limited, although some major vendors are beginning to answer to the community's demand for better activity management capabilities [25].

Many content-authoring systems appeared simultaneously with SCORM and the IMS content-packaging and metadata specifications. By contrast, the activity management side has been very slow to get off the ground, for the obvious reason that manipulating digital content in a web environment is a much easier technical proposition than modelling activities and workflows.

Since IMS-LD was presented, different tools have been created specifically to offer full support for the IMS Learning Design specification, which advocate IMS philosophy regarding the components that must be involved in the design and execution of an IMS-LD Learning unit (Figure 4).

As it can be observed in Figure 4, the design of the learning ("Design Time" section) can be effected by means of the description of elements such as roles, activities, frameworks, methods, properties, conditions and notifications, using an editor of learning units (Reload LD Editor can be used[6], Figure 5). An IMS LD editor enables the learning designer to describe all the elements that comprise the teaching-learning process, including the sequence of activities, producing a learning unit (a .zip file) which includes the imsmanifest.xml file with the final description in the XML language defined by IMS-LD.

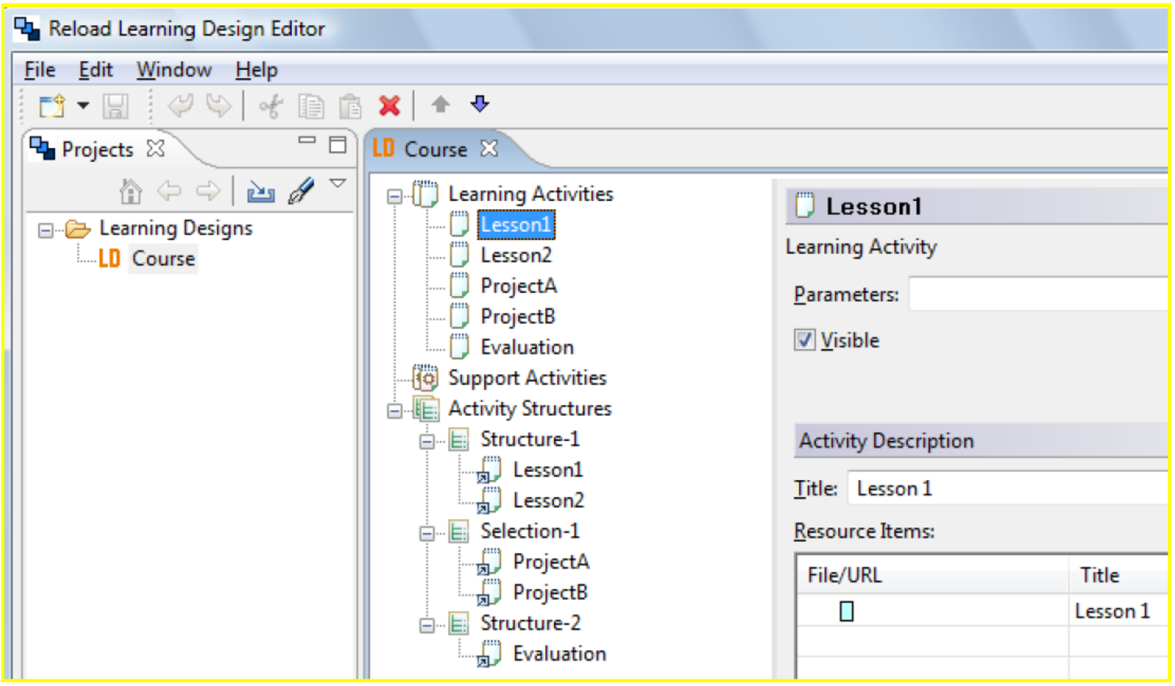


FIGURE 5. Editing a learning unit with reload LD editor

Then, it is required a run time environment to control the execution of the learning process ("Run Time" section in Figure 4), through activation of a software which is generically called the Learning Design (LD) Player. One of the most used applications is Reload LD Player[7], which is based on a Web engine named CopperCore[8], developed by the Open University of Nederland (OUNL). CopperCore is an open source IMS Learning Design Engine that supports all three levels of IMS Learning Design (A, B and C). IMS Learning Design is a complex and semantically rich specification, so it is not a small feat to provide full support [26].

Both, LD editor and LD player, must be able to handle the XML code that the learning unit includes, with the established format by the IMS-LD specification. The LD editor must generate the file from the design, whilst the LD player must parse the xml manifest

[6]http://www.reload.ac.uk/ldeditor.html
[7]http://www.reload.ac.uk/ldplayer.html
[8]http://coppercore.sourceforge.net

to control the execution of the learning units. Listing 2 shows part of the content included in the manifest of the Learning unit, the workflow of which was represented in Figure 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<imscp:manifest
  xmlns:imscp="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
  http://www.imsglobal.org/xsd/imscp_v1p1p3.xsd
  http://www.imsglobal.org/xsd/imsld_v1p0
  http://www.imsglobal.org/xsd/imsld_level_a_v1p0.xsd" identifier=
  "Curso">
  <imscp:organizations>
    <imsld:learning-design identifier="Root">
      <imsld:title>Root</imsld:title>
        <imsld:components>
          <imsld:activities>
            <imsld:learning-activity identifier="Lesson1">
              <imsld:activity-description>
              <imsld:title>Lesson 1</imsld:title>
            </imsld:activity-description>
          </imsld:learning-activity>
          <imsld:learning-activity identifier="Lesson2">
            <imsld:activity-description>
              <imsld:title>Lesson 2</imsld:title>
            </imsld:activity-description>
          </imsld:learning-activity>
          <imsld:learning-activity identifier="ProjectA">
            <imsld:activity-description>
              <imsld:title>Project A</imsld:title>
            </imsld:activity-description>
          </imsld:learning-activity>
          <imsld:learning-activity identifier="ProjectB">
          <imsld:activity-description>
            <imsld:title>Project B</imsld:title>
          </imsld:activity-description>
          </imsld:learning-activity>
          <imsld:learning-activity identifier="Evaluation">
            <imsld:activity-description>
              <imsld:title>Evaluation</imsld:title>
            </imsld:activity-description>
          </imsld:learning-activity>
          <imsld:activity-structure identifier="Structure-1" structure-
            type="sequence">
            <imsld:learning-activity-ref ref="Lesson1"/>
            <imsld:learning-activity-ref ref="Lesson2"/>
          </imsld:activity-structure>
          <imsld:activity-structure identifier="Selection-1" structure-
            type="selection">
      <imsld:learning-activity-ref ref="ProjectA"/>
      <imsld:learning-activity-ref ref="ProjectB"/>
     </imsld:activity-structure>
     <imsld:activity-structure identifier="Structure-2" structure-
      type="sequence">
      <imsld:learning-activity-ref ref="Evaluation"/>
     </imsld:activity-structure>
    </imsld:activities>
   </imsld:components>
  </imsld:learning-design>
 </imscp:organizations>
</imscp:manifest>
```

Listing 2. IMS-LD code of the activity diagram in Figure 3

Although the main tasks for edition and execution of learning units can be performed with many of the tools based on IMS-LD, such as Reload, ReCourse[9] or CopperAuthor, these tools do not offer the same set of functionalities. The main differences are related to the availability of visual utilities to model learning processes. In the absence of such functions, processes have to be modeled using textual descriptions that usually are not intuitive and uncomfortable to deal with tools that presently nowadays offer visual modeling include Learning Activity Management System (LAMS)[10], Prolix[11], MOTPlus[12] o Collage[13]. These tools are also starting to offer the possibility of reusing learning design patterns, similar to trends that can be observed in other disciplines such as software engineering. A common problem with some of these tools is that they do not offer support for the three levels that the IMS-LD specification considers[14].

4. **UML/IMS-LD Tools Interoperability Based in XSL Transformation.** It makes sense to guarantee interoperability among UML tools and IMS-LD editors if a model-driven based approach is applied to learning units designs. Such approach entails creating an abstract model using UML activity diagrams, which later are transformed into a specific model in the IMS-LD language. The format of a Learning unit in this latter model can be distributed to different Learning Management Systems.

The proposed method for the instructional design process comprises the following stages: 1) to use an UML visual design application to describe by means of an Activity Diagrams, the learning designs in terms of student and teacher's activities, 2) to export the design to an XMI format, compatible with other modelling tools, 3) to transform the activity diagram and to integrate it into the manifest of a Learning unit package that complies with the IMS-LD specification, and 4) to use an IMS-LD authoring tool (such as Reload) in order to complete the final description of the elements of the Learning unit created (Figure 6).

Figure 6 includes additional stages that consider the possibility of retrieving the visual design of an IMS-LD Learning unit to perform modifications in the learning design at an abstract level, so later it will be possible to generate a new version of the Learning unit.

Most of the editors that offer full support to the IMS-LD specification do not have a graphical design environment, thus the user should design learning units by editing text.

[9]http://tencompetence-project.bolton.ac.uk/ldauthor
[10]http://www.lamsinternational.com
[11]http://prolix-glm.sourceforge.net/
[12]http://www.licef.teluq.uquebec.ca/fr/realisations/questionnaire.htm
[13]http://sourceforge.net/projects/collage/
[14]http://www.unfold-project.net/general_resources_folder/tools/currenttools

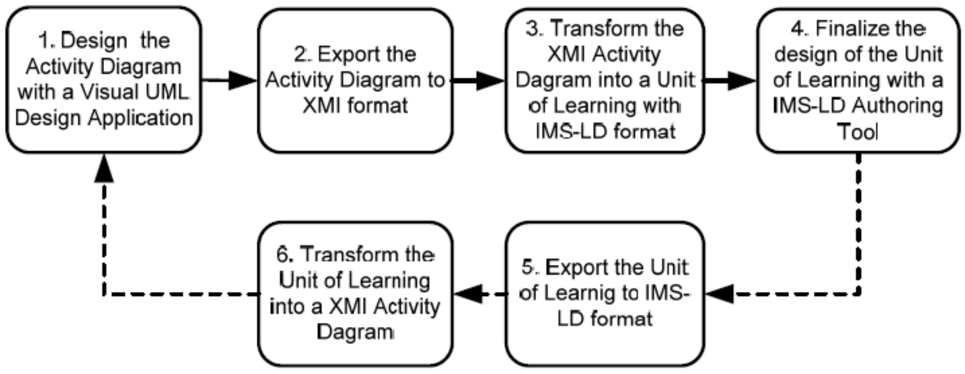

1. Design the Activity Diagram with a Visual UML Design Application
2. Export the Activity Diagram to XMI format
3. Transform the XMI Activity Dagram into a Unit of Learning with IMS-LD format
4. Finalize the design of the Unit of Learning with a IMS-LD Authoring Tool
5. Export the Unit of Learnig to IMS-LD format
6. Transform the Unit of Learning into a XMI Activity Dagram

FIGURE 6. Basic stages of an instructional design process

However, there is another option: UML tools, such as the ones described in Section 3, can be used to model the sequence of activities that comprise the learning process. IMS employs this latter approach in the examples that are included in the original IMS-LD specification.

If a tool that offers the functionality of exporting activity diagrams in XMI format is used to design learning units, it will be possible to develop a program that transforms the output model into an IMD-LD and that attaches it to the UoL (.zip) package. In this way, any LD compatible editor could open the UoL and any designer could then edit that learning design with the previously created set of activities (Figure 7, left to right).

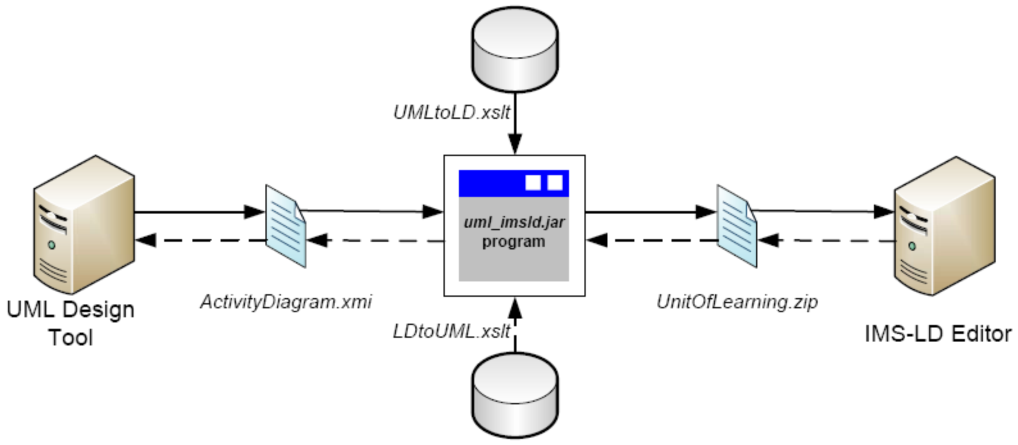
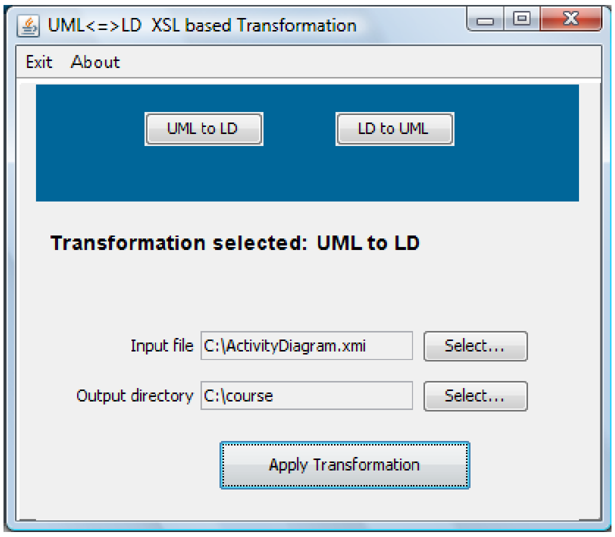


FIGURE 7. Transformation of a UML activity diagram into a IMS-LD learning unit (left to right), and generation of UML activity diagram from an IMS-LD learning unit (right to left)

It would also be very useful for that program to offer the complementary (reverse engineering) functionality: to generate an activity diagram from the contents of an existing designed learning unit. Then, it becomes possible to visualize it and to edit graphically the sequence of activities using any UML tool (Figure 8, right to left).

The authors implemented the aforementioned transformation in a Java program, called *"uml_imsld"*, which uses XSLT technology (Figure 8). XSLT is a language created by the World Wide Web Consortium for transforming XML documents into other XML documents [9]. The transformation is achieved by associating patterns with templates. First, a pattern is matched against elements in the source document and then a template is instantiated to create part of the result document. The structure of the result tree can be completely different from the structure of the source tree. In constructing the result document, elements from the source document can be filtered and reordered, and arbitrary structures can be added. A transformation expressed in XSLT is called a style-sheet and it is stored in an archive with an .xsl or .xslt extension.



Transformation selected: UML to LD

Input file C:\ActivityDiagram.xmi

Output directory C:\course

FIGURE 8. Program to transform activity diagrams into IMS-LD learning units, and to obtain activity diagrams from IMS-LD learning units

The Java source code of the program is available at http://www.cc.uah.es/hilera/software/UmlLd.zip. It presents a function for generating an IMS-LD Learning unit in a zip file from the xmi files, the name of which represents the first parameter of the function. The program uses the java SAX API[15] to process xsl transformations.

4.1. **UML-activity to IMS-LD transformation.** The process for transforming models outlined was implemented in Figure 8. The process started by using an UML modelling tool to create an activity diagram of the course that was going to be designed, and then the diagram was exported to an XMI format, generating an *ActivityDiagram.xmi* file. A code excerpt of this file, in the case of a model as in Figure 3, is shown in Listing 1. Next, using an internal XSLT transformation engine, the program *"uml_imsld"* executes the code specified in a style sheet document, generating the *UnitOfLearning.zip* file, with the *imsmanifest.xml* file in Listing 2. This latter file can be opened by any IMS-LD editor to continue the design of the learning.

Listing 3 presents a simplified extract of the transformation sheet source code (*UMLtoLD.xslt*). As it can be observed, the code loops through the input UML document locating those nodes that represent activities ("CallOperationAction") and generating the corresponding activities of the IMS-LD document in the required output format. Then, the activity structures (<activity-structure> nodes) that represent the sequence of activities are generated, using sequences or alternative paths. Due to space restrictions, the complete source code cannot be included but it is accessible at http://www.cc.uah.es/hilera/software/UmlLd.zip.

4.2. **IMS-LD to UML-activity transformation.** Reverse transformation is similar. In this case, the transformation sheet *LDtoUML.xslt*, which is complementary to the previous one, is executed by the *"uml_imsld"* program.

[15]http://www.saxproject.org

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
  xmlns:uml="http://schema.omg.org/spec/UML/2.1.2" xmi:version="2.1"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:fn="http://www.w3.org/2005/xpath-functions"
  xmlns:imscp="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <xsl:apply-templates select="xmi:XMI"/>
  </xsl:template>
```

```
<!– Transform a simple sequence of nodes –>
<xsl:template name="activity-structure-sequence" match="xmi:XMI">
  <xsl:param name="id"/>
  <xsl:param name="var_finalId"/>
  <xsl:if test="$id != $var_finalId">
    <imsld:learning-activity-ref>
      <xsl:attribute name="ref">
        <xsl:value-of select="$id"/>
      </xsl:attribute>
    </imsld:learning-activity-ref>
    <xsl:call-template name="activity-structure-sequence">
      <xsl:with-param name="id">
        <xsl:value-of select="uml:Package/packagedElement/
          packagedElement/edge[@source=$id]/@target"/>
      </xsl:with-param>
      <xsl:with-param name="var_finalId" select="$var_finalId"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

```
<!– Transform a structure with a decision and merge nodes –>
  <xsl:template name="activity-structure-selection" match="xmi:XMI">
  <xsl:param name="id"/>
  <xsl:param name="var_finalId"/>
  <xsl:param name="var_decision"/>
  <xsl:param name="var_finaldecision"/>
  ...
</xsl:template>
```

```
<xsl:template match="xmi:XMI">
          <imscp:manifest xmlns:imscp=http://www.imsglobal.org/xsd/imscp_v1p1
    xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
            http://www.imsglobal.org/xsd/imscp_v1p1p3.xsd
    http://www.imsglobal.org/xsd/imsld_v1p0
    http://www.imsglobal.org/xsd/imsld_level_a_v1p0.xsd"
    identifier="Transformation">
  <imscp:organizations>
    <imsld:learning-design>
      <xsl:attribute name="identifier">
        <xsl:value-of select="uml:Package/@xmi:id"/>
      </xsl:attribute>
      <imsld:title>
        <xsl:value-of select="uml:Package/@name"/>
      </imsld:title>
    <imsld:components>
      <imsld:activities>
        <xsl:for-each select="uml:Package/packagedElement/packagedElement/
          node[@xmi:type='uml:CallOperationAction']">
            <imsld:learning-activity>
              <xsl:attribute name="identifier">
                <xsl:value-of select="@xmi:id"/>
              </xsl:attribute>
                      <imsld:activity-description>
                <imsld:title>
                  <xsl:value-of select="@name"/>
                </imsld:title>
              </imsld:activity-description>
            </imsld:learning-activity>
          </xsl:for-each>
          <xsl:variable name="var_decision" select="count(uml:Package/
            packagedElement/packagedElement/
            node[@xmi:type='uml:DecisionNode'])"/>
          <xsl:variable name="var_InitialId" select="uml:Package/
            packagedElement/packagedElement/
            node[@xmi:type='uml:InitialNode']/@xmi:id"/>
          <xsl:variable name="finalId" select="uml:Package/packagedElement/
            packagedElement/node[@xmi:type='uml:ActivityFinalNode']/
            @xmi:id"/>
          <xsl:choose>
            <xsl:when test="$var_decision != 0">
              <xsl:variable name="decision" select="uml:Package/
                packagedElement/packagedElement/
                node[@xmi:type='uml:DecisionNode']/@xmi:id"/>
              <xsl:variable name="finaldecision" select="uml:Package/
                packagedElement/packagedElement/
                node[@xmi:type='uml:MergeNode']/@xmi:id"/>
              <xsl:call-template name="activity-structure-selection">
                <xsl:with-param name="id" select="$var_InitialId"/>
                <xsl:with-param name="var_finalId" select="$finalId"/>
                <xsl:with-param name="var_decision" select="$decision"/>
                <xsl:with-param name="var_finaldecision"
                  select="$finaldecision"/>
                </xsl:call-template>
              </xsl:when>
```

```
                  <xsl:otherwise>
                    <imsld:activity-structure>
                      <xsl:attribute name="identifier">Structure</xsl:attribute>
                            <xsl:attribute name="structure-type">sequence</xsl:att-
                              ribute>
                            <xsl:call-template name="activity-structure-sequence">
                      <xsl:with-param name="id">
                    <xsl:value-of select="uml:Package/packagedElement/
                      packagedElement/edge[@source=$var_InitialId]/@target"/>
                    </xsl:with-param>
                    <xsl:with-param name="var_finalId" select="$finalId"/>
                    </xsl:call-template>
                  </imsld:activity-structure>
                </xsl:otherwise>
              </xsl:choose>
            </imsld:activities>
          </imsld:components>
        </imsld:learning-design>
        </imscp:organizations>
        </imscp:manifest>
      </xsl:template>

< /xsl:stylesheet>
```

Listing 3. XSLT transformation sheet *UMLtoLD.xslt* for generating a IMS manifest from an activity diagram in XMI format

Listing 4 presents a short extract of the source code of this transformation sheet (*LD-toUML.xslt*). In this case, activity-structures are located in the input file and a new UML "CallOperationAction" node is generated for each activity. References to incoming and outgoing flows to other activities must also be included in the last section (edges) of the file. This generation is not a trivial matter as it requires creating local variables and using predefined functions, such as position() or last(), among others, and a special method to control the cascade levels in activity structures (sequence and selection). Due to space restrictions the complete source code cannot be included but it is accessible at http://www.cc.uah.es/hilera/software/UmlLd.zip.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:imsld="http://www.imsglobal.org/xsd/imsld_v1p0"
  xmlns:imscp="http://www.imsglobal.org/xsd/imscp_v1p1">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">
    <xsl:apply-templates select="imscp:manifest"/>
  </xsl:template>
  <xsl:template match="imscp:manifest">
    <xmi:XMI xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
      xmlns:uml="http://schema.omg.org/spec/UML/2.1.2" xmi:version="2.1">
      <uml:Package xmi:id="Root" name="Root">
```

```
          <packagedElement xmi:type="uml:Package" xmi:id="Example"
            name="Component View" visibility="public">
            <packagedElement xmi:type="uml:Activity" xmi:id="Course"
              name="Course" visibility="public">
                ... <!— global variables definition –>
                <xsl:for-each select="imscp:organizations/imsld:learning-
                  design/imsld:components/imsld:activities/imsld:activity-
                  structure">
                  ... <!— local variables definition for each act-structure –>
                    <xsl:if test="@structure-type='sequence'">
                      <xsl:for-each select="imsld:learning-activity-ref">
                         ... <!— transforming activities into a sequence of nodes
                                and edges –>
                      </xsl:for-each>
                    </xsl:if>
                    <xsl:if test="@structure-type='selection'">
                      ...
                        <xsl:for-each select="imsld:learning-activity-ref">
                          ... <!— transforming activities into decision and merge
                                nodes and edges –>
                  </xsl:for-each>
                </xsl:if>
              </xsl:for-each>
            </packagedElement>
          </packagedElement>
        </uml:Package>
      </xmi:XMI>
    </xsl:template>
  </xsl:stylesheet>
```

Listing 4. Extract of the XSLT transformation sheet *LDtoUML.xslt* for generating a XMI file from a IMS manifest

5. **Conclusions.** Business process modeling is an activity that began to be applied in organizations in the 80s to improve their efficiency. Experience in the application of this technology since then has originated standards in the field of e-learning, such as IMS-LD, that facilitates to share and reuse educational processes in the form of learning designs packaged in Learning units. Nowadays, the emergence of modeling and editing tools for learning designs (such as Reload, ReCourse, MOT and LAMS) has made this a reality.

Although there are several learning design editors that display a graphical representation of the workflow of learning units (for example LAMS), many designers have used and still use UML modeling tools to design learning actions. This is partly due to the fact that there are many free and open source UML tools, but also because UML diagrams can be exported to the standard XMI format which makes easier the diagram interchange among compliant tools.

In this article, we have presented a learning design process that starts with the use of a design UML visual application to describe, by means of Activity Diagrams, learning designs in terms of the activities that students and teachers perform; it continues with the exportation of the design into a XMI format, compatible with other UML modelling tools to finally perform the transformation of the activity diagram into a learning unit package that satisfies the IMS-LD specification, using the *uml_imsld* program. Finally, it concludes with the use of an IMS-LD authoring tool to complete the final description of the elements of the learning unit created.

In the article, we have also presented a reverse process (or re-engineering process) for automatically generating UML activity models from IMS-LD in order to obtain a graphical representation of the existing workflow of learning designs.

The main contribution of our method is that it offers the possibility of applying a real model-driven approach to design learning units, using tools based on well-known and widely accepted standards such as UML, XSLT and IMS-LD. Thus, if it is used a language such as UML which offers many supporting tools, it is possible to obtain benefits derived from the experience of use of these tools as well as those derived from the additional modelling mechanisms included; for example, the evaluation metrics, which are useful for optimizing the activity models [27]. In addition, many UML tools present the functionality of creating profiles and extensions which can be incorporated into the transformation sheets previously described. In this case no modifications to the Java application will be required.

In the literature, it is possible to find works related to different proposals of Visual Instructional Design Languages [3]; nevertheless, in the majority of the cases it is about new modeled languages that require tools or environments of instructional design developed specifically to support these languages. With this article we propose to use the widely and well known environments of modeled systems based on UML standard, for the modeled of learning processes; allowing to the developed tool the interoperability between these environments, which normally are used in software Engineering, and the learning management environments based on LD of IMS standard language, for the interchange of learning designs in the form of UML activity diagrams; which are the most adequate diagrams to represent the work flow of the learning processes, from the point of view of semiotic clarity, visual expressiveness, semantic transparency and perceived usefulness [5].

With respect to other experiences similar to the described one, we have not found free access tools like the presented one in this article. The most related work with the presented here is UML4LD, a graphical representation of abstract learning scenarios [21], where the author proposes the automatic generation of a UML activity diagram from a learning design stage using UML profiles. However, in this case, the author implemented a UML profile into an specific CASE tool, with a proprietary language and libraries in order to handle the XML-format of the learning design stages. The newness in our case is the use of a standard and well known language in the transformations, like is XSLT, that guarantees that our tool can be used for the interoperability, independent of the model environments and learning management used, as long as they satisfy respectively UML/XMI and IMS-LD standards. In addition, the complete source code that we have developed is available in the Web for its download.

The described method and tool have been applied to different university undergraduate and graduate courses, facilitating the reuse and share of learning units in a virtual university. For example, in [28], we presented a real case of application of the described work in this article, modelling a complete teaching-learning process with UML activity diagrams that are transformed in IMS Learning Design specification format, with the aim of reusing and sharing the learning design of the subjects of the "Master in Computer Science" at the University of Alcalá, through the virtual campus of this University. With every course or subject, we proceeded following these steps: 1) to create and to package learning objects to use in the subject; 2) to describe, by means of Activity Diagrams, learning design in terms of activities to be performed by students and teachers, using an UML visual design tool; 3) to export the design to an XMI format compatible with other UML modelling tools; 4) automatically, to transform activity diagrams and to integrate them into the manifest of a Learning unit package that satisfies the IMS-LD specification, using the XSLT programs described in this article; 5) to use an IMS-LD authoring tool (Reload Learning Design Editor) for the final description of the elements from the learning unit created; 6) to deploy the Learning unit in the LMS that integrate an IMS-LD player. This experience has been recently extended to other virtual undergraduate and postgraduate programs. Nevertheless our application presents a few limitations which are common to many IMS-LD edition tools. Essentially, it does not offer support for all the levels of the IMS-LD specification (A, B and C).

The authors are working to improve the software developed, so in a future version of the system it will be compliant with more than the present level A. Another extra functionality in the next version it will be the incorporation of functionalities to compute quality metrics of the activity diagrams and its application to improve learning performance [29], as well as the integration of an UML diagram viewer to visualize the activity diagram generated from a learning unit when the reverse process transformation has been applied, so that it will be no longer necessary to use an UML editor.

Acknowledgment. This research has been supported by the Government of Madrid Autonomous Community and University of Alcala (grant CCG10-UAH/TIC-5915), and by the Spanish Ministry of Education and Science (Tempros project: TIN2010-20057-C03-02). Authors also want to acknowledge support from the TIFyC research group.

REFERENCES

[1] *IMS Learning Design Specification – IMS Global Learning Consortium*, http://www.imsglobal.org/learningdesign/, 2003.

[2] E. Koper, *Modeling Units of Study from a Pedagogical Perspective: The Pedagogical Meta-model behind EML*, Open University of Netherlands, http://eml.ou.nl/introduction/articles, 2001.

[3] M. Caeiro, M. Derntl and L. Botturi, Special issue on visual instructional design languages, *Journal of Visual Languages and Computing*, vol.21, no.6, 2010.

[4] J. M. Dodero, A. Martínez and J. Torres, An extensible approach to visually editing adaptive learning activities and designs based on services, *Journal of Visual Languages and Computing*, vol.21, no.6, pp.332-346, 2010.

[5] K. Figl, M. Derntl, M. Caeiro and L. Botturi, Cognitive effectiveness of visual instructional design languages, *Journal of Visual Languages and Computing*, vol.21, no.6, pp.359-373, 2010.

[6] L. Botturi and S. Todd, *Handbook of Visual Languages for Instructional Design: Theories and Practices*, Idea Group (2007).

[7] *Unified Modeling Language (UML)*, Object Management Group, http://www.omg.org/spec/UML/, 2009.

[8] *ISO/IEC 19505-1: Information Technology – OMG Unified Modeling Language (OMG UML) Version 2.1.2 – Part 1: Infrastructure*, International Standards Organization, Geneve, Switzerland, 2009.

[9] *ISO/IEC 19505-2: Information Technology – OMG Unified Modeling Language (OMG UML) Version 2.1.2 – Part 2: Superstructure*, International Standards Organization, Geneve, Switzerland, 2009.

[10] N. Pukkhem and W. Vatanawood, An evidential reasoning approach for learning object recommendation with uncertainty, *ICIC Express Letters*, vol.4, no.3(B), pp.929-936, 2010.

[11] *XML Metadata Interchange (XMI)*, Object Management Group, http://www.omg.org/spec/XMI/, 2007.

[12] *XSL Transformations (XSLT)*, World Wide Web Consortium, http://www.w3.org/TR/xslt/, 1999.

[13] J. Yong, Internet-based e-learning workflow process, *Computer Supported Cooperative Work in Design II*, pp.516-524, 2006.

[14] J. M. Dodero, C. Tattersall, D. Burgos and R. Koper, Transformational techniques for model-driven authoring of learning designs, *Advances in Web Based Learning*, pp.230-241, 2008.

[15] N. Shimo, S. Pang, N. Kasabov and T. Yamakawa, Curiosity-driven multi-agent competitive and cooperative LDA learning, *International Journal of Innovative Computing, Information and Control*, vol.4, no.7, pp.1537-1552, 2008.

[16] D. Michael and R. Motschnig, *coUML – A Visual Language for Modeling Cooperative Environments, Handbook of Visual Languages for Instructional Design: Theories and Practices*, Idea Group, 2007.

[17] P. Laforcade, B. Zendagui and V. Barré, A domain-specific-modeling approach to support scenarios-based instructional design, *Times of Convergence, Technologies Across Learning Contexts*, pp.185-196, 2008.

[18] *IMS Content Packaging Specification*, IMS Global Learning Consortium, http://www.imsglobal.org/content/packaging/, 2007.

[19] C.-K. Ke, W.-T. Chen and M.-Y. Wu, Adaptive support of e-portfolio knowledge for student ubiquitous learning, *ICIC Express Letters*, vol.4, no.5(A), pp.1521-1528, 2010.

[20] J. M. Almendros-Jiménez and L. Iribarne, An extension of UML for the modeling of WIMP user interfaces, *Journal of Visual Languages & Computing*, vol.19, no.6, pp.695-720, 2008.

[21] P. Laforcade, Visualization of learning scenarios with UML4LD, *Journal of Learning Design*, vol.2, no.2, pp.31-42, 2001.

[22] W. Van der Vegt, O. ONeil, R. Nadolski and R. Koper, *Learning Design UML Profile*, Open University of Netherlands, http://dspace.ou.nl/handle/1820/657/, 2006.

[23] *Meta Object Facility (MOF)*, Object Management Group, http://www.omg.org/spec/MOF/, 2006.

[24] *UML Diagram Interchange (UMLDI)*, Object Management Group, http://www.omg.org/spec/UMLDI/, 2006.

[25] S. Britain, *A Review of Learning Design: Concept, Specifications and Tools, Report for the JISC E-learning Pedagogy Programme*, http://www.jisc.ac.uk/uploaded_documents/ReviewLearning Design.doc, 2004.

[26] J. R. Hilera, J. Escribano, R. Barchino, J. M. Gutiérrez, S. Otón, J. J. Martínez, J. A. Gutiérrez and L. De Marcos, An IMS-learning design player based on coppercore engine, *Proc. of e-Learning 2008*, pp.363-370, 2008.

[27] P. J. Lara, J. J. Escribano, L. Fernández and J. R. Hilera, A study of the relationship between usability and test cases precedence based on a formal model for activity diagrams, *Proc. of the Int. Conf. on Software Engineering Research and Practice*, pp.462-469, 2005.

[28] J. R. Hilera, L. Fernandez-Sanz, J. A. Gutiérrez-De-Mesa, S. Otón, R. Barchino, J. M. Gutiérrez, J. J. Martínez and L. De-Marcos, Reusing and sharing learning designs in a virtual university: A real case, *Proc. of ED-MEDIA. AACE*, pp.460-465, 2010.

[29] X. Ren, X. Zhang and K. Kyo, Quantifying the learning effect in human performance models, *International Journal of Innovative Computing, Information and Control*, vol.4, no.9, pp.2419-2430, 2008.