Authoring and Reengineering of IMS Learning Design Units of Learning

Iván Martínez-Ortiz, José-Luis Sierra, and Baltasar Fernández-Manjón, *Senior Member*, *IEEE*

Abstract—Educational Modeling Languages (EMLs) are notations that allow instructors to formally describe educational processes, including teaching and learning interactions and activities. The description of a specific teaching process using an EML is called a *learning design.* EMLs, where IMS Learning Design (IMS LD) is becoming a "de facto" standard, address aspects such as the interoperability and reusability of teaching practices across learning management systems. However, the actual application of EMLs is being hindered by different problems such as the technical skills required to use typical EMLs and the difficulty of understanding and maintaining preexisting learning designs. Thus, to promote the adoption of EMLs, it is necessary to provide more user-friendly tools and methodologies to facilitate their assimilation and reduce the workload required to use them. In this paper, we present the e-LD system, which provides a graphical notation to design or redesign learning designs, an import-modification-export process to reengineer IMS LD learning designs, and a tool to generate and analyze dependencies between different IMS LD elements.

Index Terms—Design notations and documentation, graphical notations, educational design, IMS learning design.

1 Introduction

For the last few years, e-learning has been a very active research field with real applications in industry and educational institutions. Even though e-learning has been successful in many cases, a number of limitations have been identified and have attracted criticism. One of the key issues identified is that e-learning environments are too focused on the learning content to be consumed by the learners [1]. However, an effective learning process requires more learning activities than simply being exposed to the content. It should also include other activities such as completing exercises, preparing essays, discussing topics, and assessing progress. Such activities reinforce the knowledge contained in the content. Usually, when teachers or domain experts design a course, they decide on the content to be included, the activities to be performed, and the order in which activities should occur to achieve effective learning. In other words, teachers must design a teaching method.

The definitions of these teaching methods, referred to hereafter as learning designs, include the goals and scope of the course, methods for evaluation, course materials, and the activities to be performed by the students. An explicitly written learning design can be used for different purposes. For example, it may be validated by a quality department before the course is deployed, or it may be reviewed by students before enrollment. Traditionally, this documentation task is performed by creating descriptive documents that use natural language. Nevertheless, learning designs can formally be described by using suitable Educational Modeling Languages (EMLs). The most widely extended formal EML currently used is IMS Learning Design (IMS LD) [2], [3]. In EMLs, the minimum significant educational piece is no longer the Learning Object [4] (i.e., content) but the activity or course. With IMS LD, a course is called a Unit of Learning (UoL) and includes not only content but also learning objectives, activities, and other resources.

From a pedagogical perspective, an EML is a notation system that teachers or instructors can use to formalize the learning designs that they have in mind. This formal approach, as opposed to using natural language, allows the automatic processing of these designs by a computer system. From a technical perspective, the EML can also be seen as a scripting language for Learning Management Systems (LMSs) that allows the configuration of the learning experiences in these systems. But contrary to traditional programming languages created for technical staff, the EMLs' intended target audiences are teachers and instructors.

However, the application of EMLs is not devoid of problems. A formal EML should be carefully designed to provide an adequate balance between the expressivity features related to its machine processing capabilities and high-level abstractions in order to simplify its application by humans. IMS LD, for example, is a powerful EML but its use in practice is being hindered by different problems such as the difficulty in using advanced features and the technical skills needed for its application, which are far beyond the reach of most real users who do not have mature, user-friendly supporting tools.

To address this complexity-expressiveness balance, our approach is to make a conceptual distinction between the two kinds of EMLs: exchange EMLs and authoring EMLs [5], [6]. Authoring EMLs are domain-specific languages closer to instructors' needs and ways of thinking. Because authoring EMLs are specifically adapted to instructors' expertise, authoring and repurposing tasks are far more reasonable for nontechnical instructors. Exchange EMLs are closer to the machine level, effectively becoming a low-level representation for e-learning applications, allowing

• *The authors are with the Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, c/ Profesor José García Santesmases s/n, Facultad de Informática, 28040 Madrid, Spain. E-mail: {imartinez, jlsierra, balta}@fdi.ucm.es.*

Manuscript received 16 Dec. 2008; revised 8 Mar. 2009; accepted 22 Mar. 2009; published online 27 Mar. 2009.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2008-12-0113.
Digital Object Identifier 10.1109/TLT.2009.14.

1939-1382/09/$25.00 © 2009 IEEE Published by the IEEE CS & ES

interoperability and the customization of any compliant e-learning platform to suit specific needs. In this sense, IMS LD should be classified as an exchange EML.

In our opinion, in order to promote the use of EMLs by teachers and instructors, it is necessary to provide graphical notations, which are more user-friendly than the terse XML syntax usually provided in EML specifications (in the way it is done in LAMS [7]). These notations, closer to the needs of the user, are then translated to the more machine-friendly notations of exchange EMLs via an exportation process. For this purpose, in this paper, we propose a visual language that includes concepts closely related to IMS LD, which is our target exchange EML.

The paper also addresses how to redesign pre-existing IMS LD UoLs using the aforementioned visual language. For this purpose, the approach proposes a semiautomatic process according to which an initial UoL in the authoring EML is produced, as well as a report that helps instructors to complete the redesign. In particular, we propose to generate this report based on a thorough analysis of the dependencies between elements that can affect the runtime behavior of the original UoL.

This paper is structured as follows: In Section 2, we analyze the IMS Learning Design specification and motivate our approach. Section 3 describes the e-LD approach, which is the whole approach promoted in this paper. The authoring process of a new UoL is described in Section 4. The reengineering process of an IMS LD UoL is presented in Section 5. Section 6 analyzes some related work and compares it to that reported in this paper. Finally, Section 7 provides some conclusions and future lines of work.

2 IMS Learning Design Analysis

IMS Learning Design [2], or simply IMS LD, is a specification to represent and encode e-learning courses. Furthermore, IMS LD is focused on the design of pedagogical methods that manage learning activities linked to learning objects within a learning flow [3]. This learning flow consists of *plays*, *acts*, *role-parts*, and *simple* and *structured activities*, and is flexible enough to provide several personalized itineraries, depending on the role assigned or a set of rules.

IMS LD uses the term *Unit of Learning* (UoL) to represent the minimum significant educational piece. According to IMS LD, a UoL includes content, learning objectives, activities, and other resources.

To facilitate the adoption of IMS LD, the specification is divided into three levels:

- **Level A:** Specifies core components using concepts like *method*, *play*, *act*, *role*, *role-part*, *learning activity*, *support activity*, *activity structure*, and *environment*.
- **Level B:** Adds *properties* and *conditions* used for the dynamic personalization of a UoL based on the previous knowledge of the learner and learner performance within the UoL.
- **Level C:** Adds *notifications* allowing the communication between actors involved in the UoL as well as providing a new event-based mechanism for UoL personalization.

In addition to this layered organization of the specification, any IMS LD UoL is clearly divided into two different parts: the *static* and *dynamic* parts. The static part comprises the definitions of *activities* (simple and structured), the *environments* within the activities are carried out, and the participants' *roles* involved in the UoL. The dynamic part includes the sequencing of the simple and structured activities defined in the static part and the assignment of activities to the different participants.

The following sections conduct an analysis of IMS LD and the sequencing expressiveness of activities. This analysis reveals some of the authoring complexities of an IMS LD UoL that have direct impact on our approach. This analysis is focused on Levels A and B, as the work reported in this paper does not provide support of IMS LD level C.

2.1 Analysis of IMS LD Level A Sequencing Expressiveness

To understand the activities sequencing expressiveness of IMS LD Level A, it is necessary to understand how the UoLs' participants (i.e., students or instructors) perceive the sequencing of activities.

Both IMS LD specification [2] and existing IMS LD players (e.g., SLeD [8], GRAIL [9], [10]) provide participants with a tree-based user interface to interact with the activities of the UoL. This activity tree is modified as a result of the participant's interaction with the UoL, where new elements become *visible* after the participant *completes* the interaction with some of the visible activities in the activity tree. Therefore, activity sequencing in IMS LD is closely related to these two key concepts: *visibility* and *completion*.

Visibility of elements is controlled in IMS LD level A through the attribute *isvisible*. At design time, some Level A elements (e.g., *learning activity*, *support activity*, etc.) may set the *isvisible* attribute to *false*, making these elements hidden to the participants at runtime. IMS LD Level A only provides a mechanism to make simple hidden activities visible, but there is no mechanism for the *play* element.

Completion of IMS LD elements serves two purposes. It serves as a simple monitoring mechanism for participants' progress and it affects the sequencing of the UoL's other elements.

Each element defined in the method section of the UoL can be completed, as can activities (both simple and structured ones). IMS LD Level A is equipped with four mechanisms to decide when one of these elements has been completed: completed by default, user choice, time-based, and based on included elements' completion.

Note that simple and structured activities can be used in different parts of the same UoL and their completion and visibility state is shared among all references to the same activity. That is, if an activity is completed, all its occurrences are also completed. Similarly, if an activity becomes visible, all of its occurrences subsequently become visible.

The sequencing of the activities in IMS LD Level A is done by using IMS LD elements: *method*, *play*, *act*, *role-part*, and *activity structure* (*activity structures* are of two types: *sequence* and *selection*). These elements act as building blocks that are aggregated to define the entire sequencing behavior. All of these elements have a predefined sequencing behavior (defined in the IMS LD Information Model [2]) as follows:

TABLE 1
IMS LD Level B Properties Applicability

Property type	IMS LD element	Scope	Owner	Intended Purposes
Local	*loc-property*	U	U	• Parameterization of UoL run.
Role	*locrole-property*	U	R	• Role intercommunication. • Role synchronization.
Personal	*locpers-property*	U	P	• Store participants' progress. • Participant communication.
	globpers-property	S	P	• Store participants' personal profile. • Store participants' learning records.
Global	*glob-property*	S	S	• Organization data (e.g. name). • Organization policies.

Scope: UoL run (U), System (S). Owner: UoL run (U), Participant (P), System (S), Role (R)

- **Method:** *Methods* run in parallel all the *plays*, which they include.
- **Play:** *Plays* run in sequence all the *acts*, which they contain. That is, an act starts after the previous act has been completed.
- **Act:** *Acts* run in parallel all the *role parts*, which they include.
- **Role part:** A *role part* runs the activity that it refers to.
- **Activity structure (sequence type):** Runs in sequence all the activities included. That is, an activity starts after the previous activity has been completed.
- **Activity structure (selection type):** In these activity structures, participants can randomly choose the order in which they perform the activities included.

2.2 Analysis of IMS LD Level B Enhanced Sequencing Expressiveness

IMS LD Level B introduces more powerful sequencing concepts, which are particularly useful in allowing a detailed personalization of the learning flow and in supporting different pedagogical approaches [11].

The new elements related to activities sequencing introduced in IMS LD Level B include *properties*, *conditions*, and *enhanced completion* mechanisms.

Properties provide a storage mechanism to store pieces of information of different types (e.g., numbers, text, etc.) similar to variables in programming languages. The *scope* of a property defines the context within which it is defined. Moreover, properties can be grouped through the concept of *property group* to simplify management.

In a similar way to the deductive approach proposed by Heyer et al. [12] and the inductive approach put forth by Koper and Burgos [11], we have followed a deductive-inductive approach to analyze the applicability of the different types of properties available in IMS LD level B. This analysis is based on the study of the available IMS LD level B UoLs at Learning Networks DSpace repository[1] and a careful study of the IMS LD specification. Table 1 summarizes this analysis.

The second key concept concerning activities sequencing in IMS LD Level B are *conditions*. Conditions are defined as a set of reactive *if-then-else* rules. A rule is guarded by an expression so that when the expression holds, the actions included in the rule's *then* section are executed; otherwise, the actions in the rule's *else* section are executed. Guard expressions are evaluated whenever an important event happens during the execution of the UoL, particularly the change of a property's value. Available actions in IMS LD allow for: 1) the personalization of the learning flow through visibility modifications of activities and plays, 2) personalization of the learning content, and 3) modification of the properties' values.

In a similar way to the property analysis done in Table 1, we have analyzed the rules of the conditions. As a result, we propose the classification summarized in Table 2.

Properties and conditions in IMS LD level B enhance the completion mechanism, introducing the possibility of completing an element based on a property's value, and particularly for *acts*, based on Boolean expressions.

The sequencing features of IMS LD levels A and B, especially conditions, provide a powerful and flexible mechanism to create customizable learning flows. However, this great expressiveness also introduces high complexity during the UoL authoring. Moreover, to reuse a UoL that makes extensive use of conditions for personalization requires thorough knowledge of conditions, and thus, becomes a complex task. The e-LD approach addresses these authoring and reuse issues in IMS LD UoL (levels A and B).

3 The e-LD Approach

In our opinion, one of the issues that hinders the adoption of EMLs, in general, and of IMS LD, in particular, is the lack of user-friendly supporting tools that simplify the use of these languages.

Our aim with the e-LD approach is to provide an evolving authoring tool, called e-LD Author, and a set of support tools for helping instructors in the creation of their courses. e-LD Author plays the same role as Computer-Aided Software Engineering (CASE) tools in Software Engineering processes.

In our e-LD approach, we have fully considered the traditional Analysis, Design, Development, Implementation, and Evaluation (ADDIE) approach [13] widely used in the development process for the creation of instructional courses. This same development process can be applied to

1. http://dspace.learningnetworks.org.

TABLE 2
IMS LD Level B Conditions' Rules Classification

Rules types	Requisites	Intended applicability
UoL adaptation	Rule's expression only references: loc-property, globpers-property and glob-property properties. Rule's actions modify the visibility of activities and/or plays.	Provides global adaptation (flow and learning content) of the UoL based on the learner profile (e.g. accessibility, preferred learning style, previous knowledge).
Participants' flow adaptation	Rule's expression particularly uses locpers-properties. Rule's actions modify the visibility of activities and/or plays. Rule's actions modify a property related to the completion of an element.	Modifies the learning flow based on the performance of the participant in the UoL's run.
Roles' flow adaptation	Rule's expression particularly uses locrole-properties. Rule's actions modify the visibility of activities and/or plays. Rule's actions modify a property related to the completion of an element.	Modifies the learning flow of participants in a specific role. Particularly, rules of this type are used to provide synchronization at role level.
Expression calculations	Rule's actions modify properties referenced in other rules.	Provides a simplification mechanism for complex expressions, allowing the factoring of complex expressions.
Content personalization	Rule's actions modify the available learning resources in an environment and/or adapt learning contents tagged as *imsldcontent*.	Personalization of content and resources based on the participant's profile, assigned responsibilities in the UoL and performance within the UoL run.

the creation of UoLs (see [14]). The rest of this section briefly introduces the five phases of the ADDIE process and how they are related to our approach, and also outlines some final remarks.

The *analysis* phase comprises the identification of learning needs, the audience's needs, the audience's knowledge, and any already available content (e.g., available UoLs). Once the learning needs are identified, the outcome of this phase is a set of goals and objectives/competencies. e-LD Author offers a graphical notation to describe learning objectives (see Section 4.2) and their relationships. e-LD also provides support for the analysis of a preexisting UoL (see Section 5), allowing instructors to evaluate whether it meets some of the objectives of the course's learning design, therefore becoming a candidate for total or partial reuse.

The *design* phase comprises the overall learning design process including the selection of activities (e.g., quizzes, exercises, etc.), learning contents and tools (e.g., media, communication) needed to accomplish the identified learning objectives. In addition, the sequencing of these activities is also defined during this phase. The outcome of this phase is the instructional design document. This document represents the instructor's learning design, simply providing a high-level overview of the entire UoL without containing any actual content.

The application of EMLs fits in this design phase. Instead of creating an informal document using natural language or filling in a template, the instructional design document is created by using a particular EML that produces a formal document, which allows for its automation (i.e., interpretation in an EML player).

The aim of the e-LD Author is to allow instructors to design documents compliant with IMS LD during the creation of educational programs. But as previously mentioned in Section 1 and in the analysis performed in Section 2, IMS LD's authoring becomes complex due to its great expressiveness. We address this issue by proposing a collaborative process (see [5] for details) between developers and instructors to create an *authoring EML*. This authoring EML provides enough expressiveness required by instructors to design their courses. Compatibility with an *exchange EML*, in this particular case IMS LD, will be maintained by an exportation procedure.

Flowchart notation was one of the very first techniques used to describe business processes [15]. In addition, flowcharting has been extensively used in computer science to describe algorithms in an abstract and programming language-independent way. We therefore believe that a flow-oriented language can serve as the basis for authoring EMLs. Although the authoring EML definition is independent of its notations (e.g., XML, textual, graphical), graphical notations stand out among other notations because they provide powerful help for describing and understanding complex systems [16]. In this respect, the e-LD Author tool provides instructors with a visual notation for the flow-oriented authoring EML, used to define the activities and their sequencing (see Section 4). Note that the authoring EML provision process can be performed in parallel to the UoL authoring process.

The *development* phase is where the actual creation of learning materials is carried out. e-LD Author currently does not include specific tools to author learning contents, but it provides support for importing learning contents that

conform to IMS Content Packaging (IMS CP) [17] specification and SCORM specification [18].

The *implementation* phase is where the UoL is put into action. In addition to the translation process from the authoring EML to the exchange EML, the technical requirements of an IMS LD's UoL has to be tackled, that is, the construction of a valid IMS CP and the publication of the UoL in an IMS LD Player. e-LD Author provides support for these three tasks. Using this support, the author can export the new UoL to an IMS CP and then publish this UoL in an IMS LD Player (currently only CopperCore has been tested).

The *evaluation* phase represents a final review checkpoint for the UoL. This phase consists of: 1) formative and 2) summative evaluation. Although formative evaluation should be performed by peers at each stage of the ADDIE process, it is also easy to collect learners' opinions about the course through a short survey immediately after the course is completed. This information can be used to improve the UoL. Summative evaluation consists of tests designed to determine the effectiveness of the UoL against established objectives. e-LD Author does not currently include a tool to create questionnaires and tests; however, instructors can create them by using a specific authoring tool and then link them as a course resource.

It is important to note that the ADDIE process does not impose any pedagogical approach during the authoring of the UoL. The instructor is therefore responsible for the appropriate selection of the pedagogical methodology and for designing the UoL accordingly. The application of EMLs provides some advantages, for example, UoLs formalized with an EML are less prone to error than a UoL described using natural language, due to the elements of EML having well-defined semantics. Moreover, EMLs also allow for the formalization of proven teaching practices and pedagogical methodologies such as UoLs' skeletons/template that can later be personalized.

Whether the creation of a UoL starts from a completed UoL previously produced or from a template, it is necessary to provide instructors with support for this task. The following section describes how e-LD addresses this need.

4 Authoring of IMS LD UoL: The Visual Notation

The use of graphical notations to provide a visual syntax for modeling languages has been tested and put into practice in many different domains. Some examples include databases with Entity-Relationship models for defining database schemas [19], software engineering with Unified Modeling Language (UML) for describing software systems, and business applications with Business Process Management Notation (BPMN) for describing business processes [20].

UML and BPMN specifications not only introduce graphical notations for software artifacts or process descriptions but also define abstract metamodels for these domains. In a similar way, we have defined an abstract metamodel and a particular graphical notation for this metamodel (presented throughout the section). However, because of this separation between abstract model and notation, it is possible to define different notations for the same metamodel (e.g., textual, XML, etc.) or even more, to allow the personalization of the graphical notation based on user preferences.

Graphical notations have been developed to reduce the cognitive load when working with complex semantic models. They provide a simpler notation that can be more clearly understood by a wide range of users, from technical to nontechnical staff. Following this trend, we propose the use of graphical notations for the design of UoLs [21]. These notations include:

- A notation for participants' *roles*. This notation provides instructors with a form of notation for defining the different participant types (called roles). These roles are used to classify the participants in different types (e.g., student, teacher) and to assign responsibilities to the different roles defined.
- A notation for *learning objectives*. This notation allows instructors to define which goals (learning objectives) will be covered in the UoL. For these purposes, the instructor can define a high-level goal as the overall objective and, later on, refine this objective into subobjectives to be achieved by the different parts of the UoL. With this type of notation, it is also possible to define the participants involved in reaching these goals.
- A notation for *defining activities*. This notation contemplates the definition of the different activities to be performed during the execution of the UoL. Using this notation, instructors analyze which activities are needed to achieve the learning objectives. They then design activities describing what is to be done and which tools (chat, dossier, laboratory tool, etc.) should be used. These activities also include the instructions and the resources (learning contents and tools) needed to perform the activities. Activities can be classified into simple and structured ones. Structured activities aggregate simple activities by adding an implicit runtime behavior. As structured activities can be very large and complex, the notation introduces hierarchical abstraction facilities.
- A notation to describe the sequencing of *activities*. By using this notation, instructors make the learning flow explicit through different activities that comprise the UoL. In addition, the notation allows the definition of roles involved during the performance of the activities. Sequencing definitions can be a simple ordering of activities applied to all participants or they can provide a personalized definition of the learning flow based on the performance of the UoL's participant. This definition itself can be verbose. Therefore, the notation also introduces hierarchical decomposition mechanisms. Composite elements can be collapsed to hide some parts of the model that can also be refined in a separate diagram.

All these notations coexist in a unified, flow-oriented, view of the learning design integrating all of the design aspects. This feature, together with a user-friendly visual representation, should increase the usability of the notations with respect to more general exchange-oriented ones (such as IMS LD). This flow-oriented graphical notation is

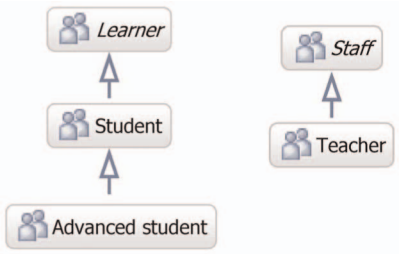

Fig. 1. Example of a role hierarchy definition using the graphical notation. Arrows represent the *is a* relationship between roles (e.g., *Teacher* is an specialization of Staff).

used within the e-LD author tool to define learning designs. Once instructors create their learning designs, they can automatically export these learning designs to create UoLs compatible with the IMS LD specification [22].

The following sections go inside the different notations that constitute the unified flow-oriented view.

4.1 Notation for Describing Roles

UoL participants' roles are defined in a diagram that represents the hierarchy of the roles. Each role definition not only includes the role name but also specific IMS LD value attributes relevant from the author's point of view (e.g., *create-new* and *href*). In addition, a more verbose textual description of the role can be attached to the graphical notation.

UoL's roles are arranged into a hierarchy where the two IMS LD base roles, *Learner* and *Staff*, are used as hierarchy root elements. The *Learner* role is a base role for roles representing the different types of students in the UoL (e.g., student, fellow student, group leader, etc.). The *Staff* role is a base role for roles representing different types of faculty and staff involved in the UoL (e.g., instructor, teaching assistant, lecturer, invigilator, etc.).

The role hierarchy is a hierarchy with inheritance. That is, each subrole has the responsibilities of its superrole plus additional responsibilities assigned to the specific subrole (i.e., subroles are specializations).

Fig. 1 depicts an example of a role hierarchy definition. *Student* and *Advanced student* roles (represented by rounded boxes) are two subtypes of *Learner* where *Advanced student* is a particular subtype of *Student*. Finally there is only one *Staff* role, the *Teacher* role.

4.2 Notation for Describing Learning Objectives

UoL learning objectives are defined in a diagram. These learning objectives can be decomposed into simpler objectives. This allows for a definition of fine-grained objectives, which also represents their interrelationships. Each learning objective definition includes a verbose textual description attached to the graphical representation.

The purpose of this decomposition is twofold:

1. To allow instructors to organize and document their mental processes.
2. To provide instructors with a simple verification mechanism to validate if the learning objectives are covered by the activities of the UoL.

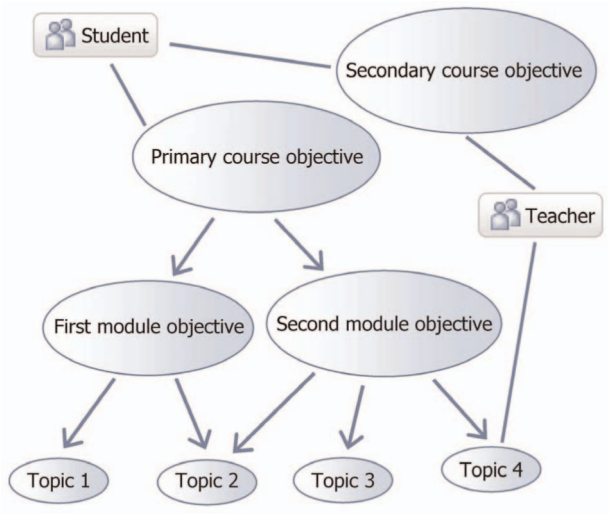

Fig. 2. Example of a learning objectives definition with role association.

Learning objectives can be associated to the different roles of participants previously defined. Fig. 2 shows an example of this notation. The example contains two objectives, *Primary course objective* and *Secondary course objective*, represented with ellipses. *Primary course objective* is subdivided into a graph of subobjectives. Arrows in the learning objectives diagram represent an inclusion relationship. Thus, *First module objective* and *Second module objective* learning objectives are part of (are included in) the main *Primary course objective* learning objective. This means that to achieve the *Primary course objective,* the minimum achievement of the *First module objective* and *Second module objective* is needed.

Moreover, two different roles participate in the achievement of the example's learning objectives: the *Student* (a learner role) and *Teacher* (a staff role) represented by rounded boxes. These roles are associated with learning objectives by using lines between the role and the objective. Note that the association between actors and objectives is inherited by subobjectives, which means that the role associated with the main objective also participates in the achievement of the subobjectives. In our example, the *Teacher* role is only involved in the achievement of *Topic 4* and *Secondary course objective* learning objectives, while the *Student* role is involved in *Primary course objective*, *Secondary course objective,* and in all the sub-objectives of *Primary course objective* (i.e., *First module objective*, *Second module objective*, *Topic 1*, *Topic 2*, *Topic 3*, and *Topic 4*). It is also possible to design learning objectives for different learner roles because the notation for learning objectives is not restricted to only one learner role per diagram.

4.3 Notation for Defining Activities

UoL graphical notation for defining activities is presented in Fig. 3. Using IMS LD terminology, the notation introduces two kinds of activities, simple and structured.

- *Simple activities* include *learning activities* (Fig. 3a), which are typically performed by students, and *support activities* (Fig. 3b), which are performed by a supporting role, usually an instructor.

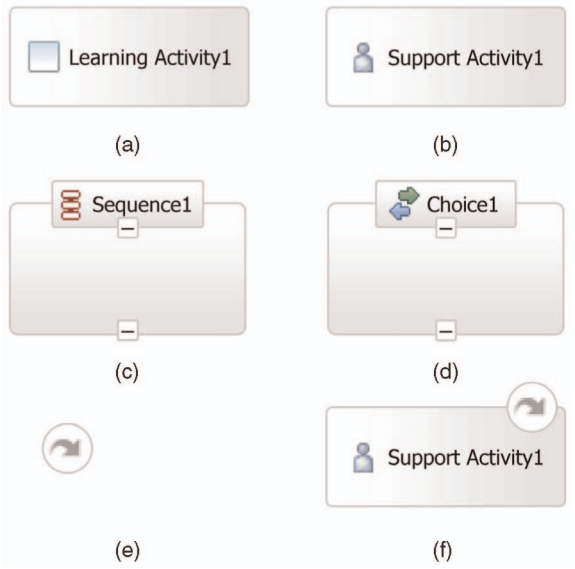

Fig. 3. Visual notation for simple and structured activities.

- *Structured activities* allow instructors to aggregate activities (both simple and structured) thus adding an implicit sequencing behavior. With these activities there are two possibilities: activities are performed in sequence or activities can randomly be chosen. According to these possibilities, there are two different graphical notations for activity structures: Fig. 3c shows the appearance of a structured activity with a fixed sequenced runtime behavior and Fig. 3d shows the appearance of a structured activity with a user selection behavior.

In an UoL, it is possible to use the same activity (simple or structured) in different places. As introduced before, the completion state of an IMS LD activity is shared among all its occurrences. Hence, a decoration symbol (a circular button with an arrow, see Fig. 3e) is automatically applied to all of the activity occurrences to highlight this behavior. Fig. 3f shows an example of a learning activity with this symbol. The authoring tool allows the instructor to *clone* an activity in order to avoid this state-sharing behavior.

Due to the hierarchical nature of activity structures, the graphical notation proposed also allows activities to be collapsed or expanded in a diagram to increase the diagram's legibility as well as the definition of the activity structures in separate diagrams.

All activities, whether simple or structured, have a set of nonvisual properties that allow the complete definition of the activity. For example, this activity definition can include the resources available at runtime or the description of the learning objectives to be achieved.

4.4 Sequencing Notation

Sequencing diagrams are very similar to UML Activity diagrams. However, in contrast to Laforcade's work [23], we are not attempting to use UML as a learning design notation, but simply to use activity diagrams as a natural choice in a flow-oriented modeling domain. Our notation also includes many other elements and concepts, which have nothing to do with standard UML.

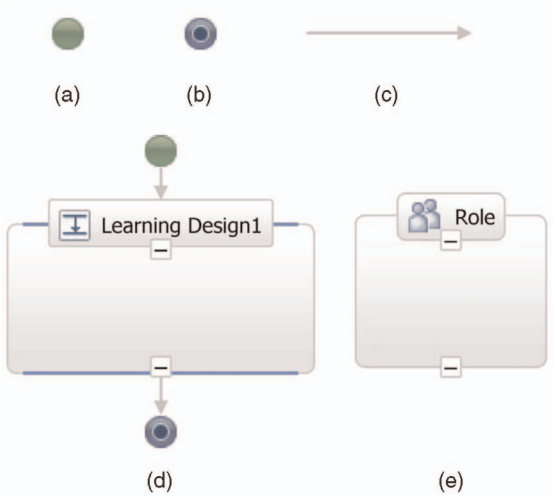

Fig. 4. Notations used in sequencing diagrams.

Fig. 4 depicts a repertory of symbols used in the e-LD sequencing diagrams. There is, therefore, a notation for indicating the following:

- The start (Fig. 4a) and end (Fig. 4b) points of the diagram.
- The different learning flows, represented by arrows (Fig. 4c).
- Parallel execution of elements is represented using rounded boxes with highlighted parallel bars at the top and bottom of the boxes (Fig. 4d).
- Assignment of participant roles to activities (Fig. 4e). Activity sequencing diagrams can be partitioned to specify which roles will be involved in the activity's performance.

Activities, whether simple or structured, will appear in activity sequencing diagrams using the same notation as described in Section 4.3.

To maintain compatibility with IMS LD, the notation for sequencing diagrams includes constructions to represent the *method* (Fig. 5a), *play* (Fig. 5b), *act* (Fig. 5c), and *role-part* (Fig. 5d). These constructions have the same behavior as IMS LD from the point of view of sequencing:

1. *Play*s inside the *method* run in parallel.
2. *Act*s inside a *Play* run in sequence.
3. *Role-part*s inside an *Act* run in parallel.

Note that, as with structured activities, the notation for the method, play, act, and role-part concepts can be collapsed and expanded as needed and its definition can be done in a separate diagram.

4.4.1 Advanced Sequencing: Personalization of the Learning Flow

To facilitate the personalization of the learning flow based on the evolution of the learner during the performance of the UoL, e-LD's sequencing notation incorporates conditions. Conditions are also present in mainstream exchange languages, such as IMS LD. However, the way of incorporating conditions in e-LD is radically different from IMS LD. In e-LD, conditions are integrated into the learning flow.

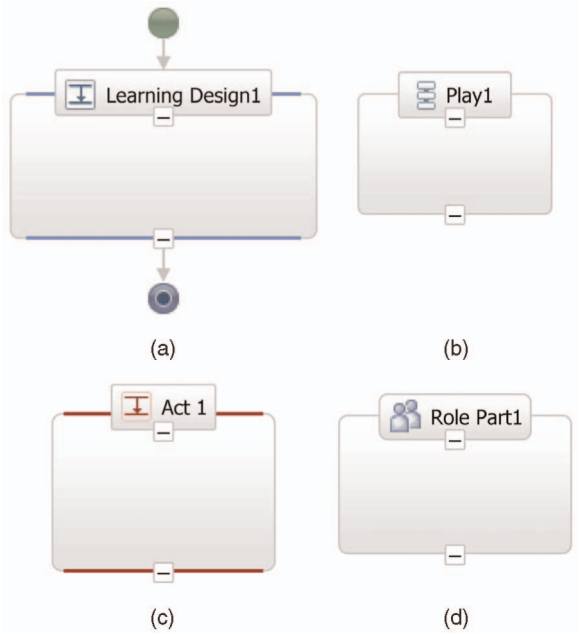

Fig. 5. Notation for method, play, act, and role-part IMS LD concepts.

This integration increases the usability of the notation, since now instructors are not required to think about a disaggregated set of rules defined in another portion of the design. They only need to reason out each relevant point in the natural evolution of a learning flow, which is a far lower cognitive load.

Fig. 6 depicts the graphical notation for conditions as follows:

- Fig. 6a shows an example of an *if-then-else* condition rule definition. This condition notation has an attached Boolean expression evaluated only when the learning flow reaches the condition. If the expression evaluates to *true*, the learning flow goes through the *If* path, and if the expression evaluates to *false*, the control flow goes through the *Else* path.
- Fig. 6b shows an example of a condition definition, where *if-then-else* rules are nested, supporting a detailed learning flow definition.
- Fig. 6c shows an example of a condition, which only contains the *If* part. This means that until the expression becomes true, the learning flow stays at the condition element.

Fig. 7a depicts an example of the graphical notation for conditions. The example scenario involves two activities, *Activity 1* and *Activity 2*, which will be performed by a student. *Activity 2* only becomes visible after *Activity 1* has been completed and when the condition of the *If1* becomes true.

Fig. 7b provides an excerpt of the equivalent IMS LD XML representation that encodes the behavior described for the graphical notation depicted in Fig. 7a, which is illustrative of the economy of the graphical notation.

The next section continues with the Geo-Quiz 3 UoL example showing the application of the visual notation described throughout this section.

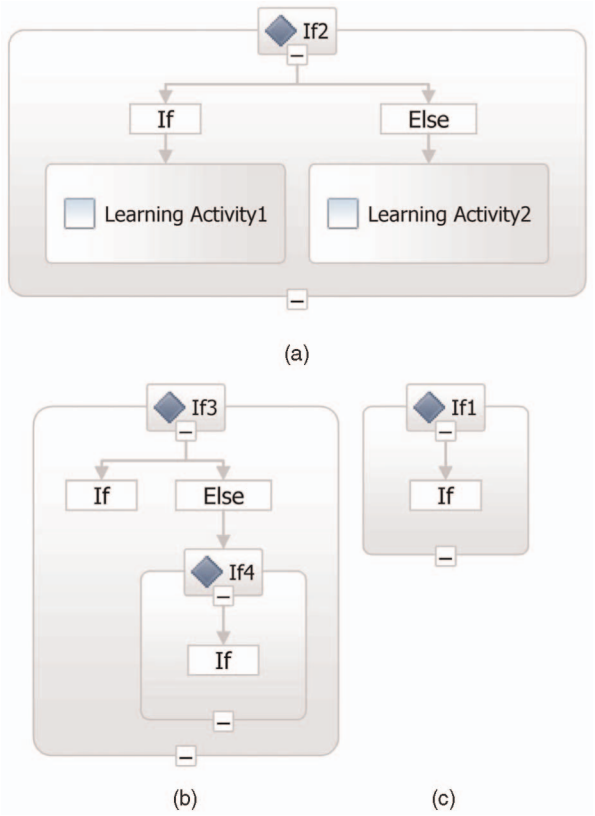

Fig. 6. Notation for conditional sequencing.

4.5 Visual Notation for the Geo-Quiz 3 UoL

The Geo-Quiz 3 UoL is one of the examples developed by Professor Koper's group as an example of conditional text and the monitoring of properties. This UoL simulates a small learning module with adaptive content and adaptive learning flow depending on the grade obtained by the student after taking a test (see [11] and [24] for a detailed explanation).

As introduced in Section 3 and explained in Section 5, the importation process produces an e-LD UoL that needs to be manually refined later, taking into account the information provided by the dependency graph (see Section 5.3). During this refinement process, the visual notation is used to fully reconstruct the semantics of the original UoL.

Fig. 8 depicts an excerpt of the complete diagram representing the sequencing of the different activities involved in this UoL. Although some parts of the sequencing of the UoL have been collapsed to create a snapshot fitting the paper's dimensions, it is much simpler to understand the structure of the UoL by simply looking at the diagram.

This simplicity is also helpful during the importation process, where the automatically generated part of the diagram (without the conditional blocks) provides a high-level view complementary to the low-level view provided by the dependency graph.

5 Reengineering of IMS LD UoL: The Importation Process

This section describes the reengineering process of a preexisting IMS LD UoL. In contrast to the authoring

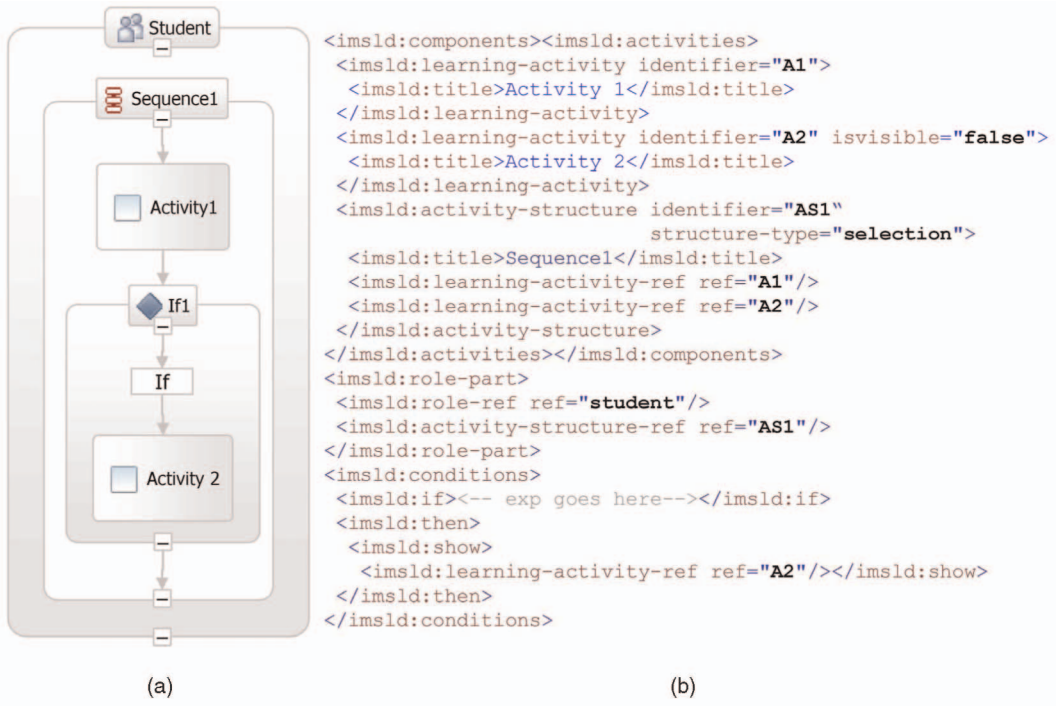

Fig. 7. Example of notation and IMS LD XML excerpt.

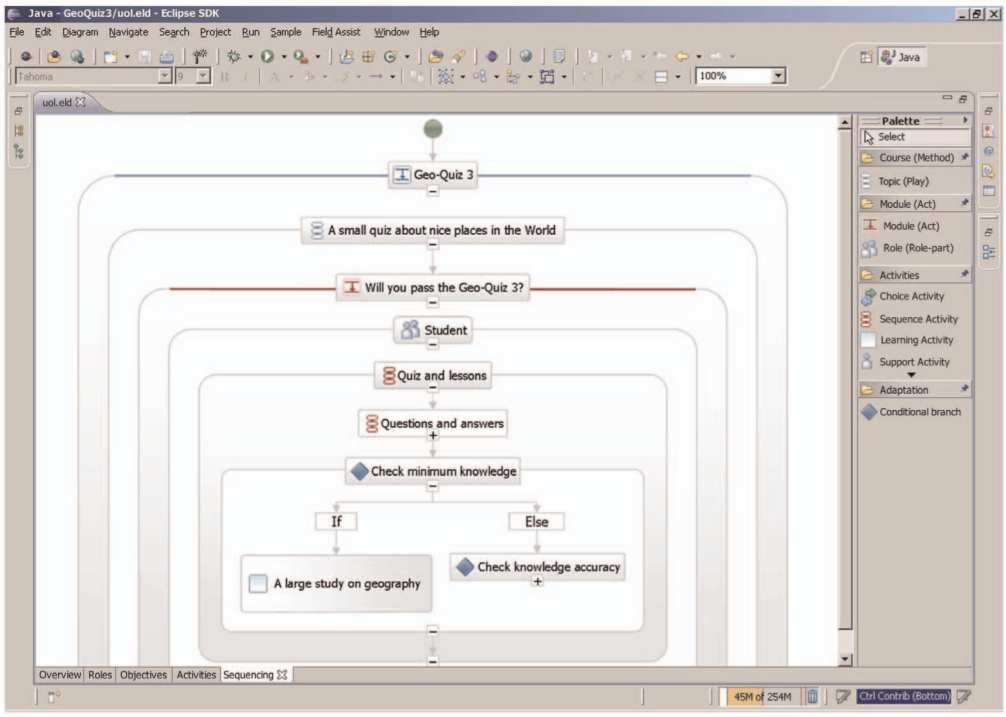

Fig. 8. Excerpt of the graphical notation for the Geo-Quiz 3 UoL.

process where instructors work on their own, the reengineering process is a collaborative process, where instructors are supported by developers.

In this section, the UoL e-LD reengineering process is analyzed from a task and resulting products perspective (Section 5.1). The UoL importation process is then detailed (Section 5.2) and, finally, an example of the dependency graph generated during the importation process is presented (Section 5.3).

5.1 Products and Tasks in the UoL Reengineering Process

Fig. 9 depicts the overall UoL reengineering process. The *requirements analysis* task involves the initial conceptualization of the UoL, identifying the roles of the participants involved (e.g., instructor, student, mediator, etc.) and the learning objectives of the UoL as a product of this task. Instructors are in charge of this task.

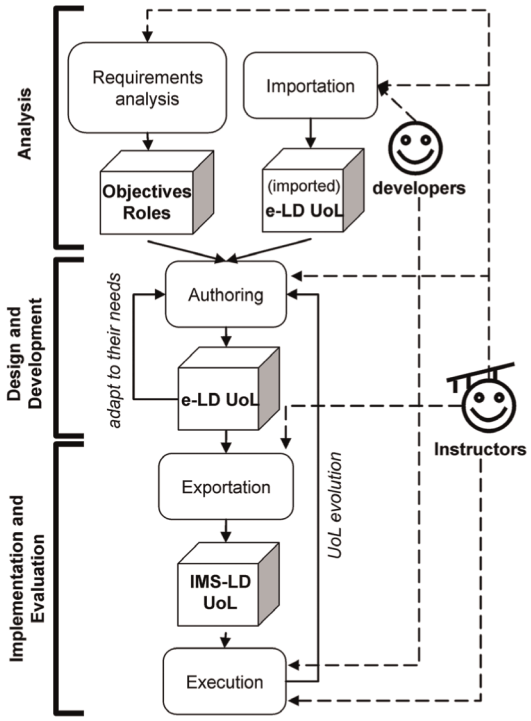

Fig. 9. Products and tasks diagram of the e-LD reengineering process. Rounded boxes represent tasks and 3D boxes represent the resulting task products. Dotted arrows represent actors' participation in each task.

The *Importation* task comprises the importation of a previously created IMS LD UoL into the e-LD Author tool. During this task, both developers and instructors are involved. Instructors perform this task as part of the analysis phase in the ADDIE process and they are the leaders of this task, whereas developers assess instructors regarding interpretation of the semantics of IMS LD and provide support regarding the programming skills needed to import IMS LD level B UoLs. As a result of this task, an UoL formalized using the authoring EML is created. Because of this task's complexity, the detailed explanation is given in Section 5.2.

The *Authoring* task is where instructors check and adapt the (imported) UoL to fulfill the learning objectives identified. This task may need several authoring sessions, usually after a formative UoL evaluation, to refine the design of the UoL. An almost-ready-to-execute UoL is the result of this task.

The *Exportation* task takes as input the UoL generated in the Authoring task to export it to a new UoL compliant with the IMS LD specification. The translation process tackles with the differences between the authoring and exchange EMLs. This process is carried out automatically by the e-LD Author tool, generating a content package that is ready to be run in an IMS LD player.

Finally, the *Execution* task is where instructors publish the UoL created, configure the IMS LD player (e.g., participants' enrollment) with the support of developers, and finally, create an instance of the UoL. Participants' feedback during UoL execution may be used to evolve the UoL into new UoL instances.

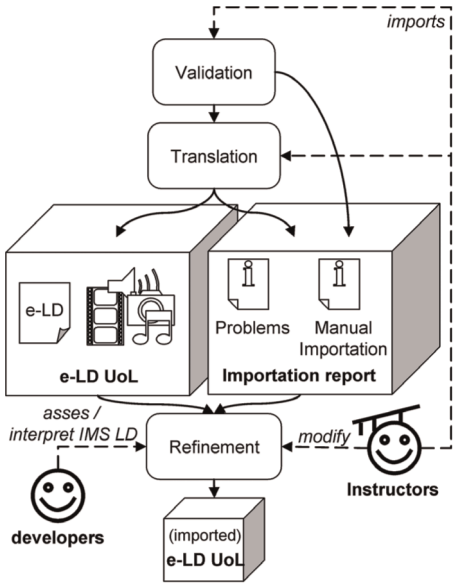

Fig. 10. Detailed importation process of an IMS LD UoL. Rounded boxes represent tasks and 3D boxes represent the resulting task products. Dotted arrows represent the participants' involvement in each task.

5.2 The IMS LD UoL Importation Process

The IMS LD UoL importation process addresses the gap between the exchange EML (IMS LD) and the flow-oriented authoring EML (graphical notation presented in Section 4). The importation process is sketched in Fig. 10.

The *Validation* task involves several steps toward a complete validation of the importing UoL. Particularly, the validation comprises three groups of tests:

1. IMS CP Package Information Format (PIF) tests.
2. IMS CP *imsmanifest.xml* tests.
3. IMS LD UoL tests.

The first group of tests is related to IMS CP requirements regarding the packaging format (i.e., a zip file) and the existence of an XML file named *imsmanifest.xml* (hereafter the *manifest*) along with control files (e.g., Document Type Definition or XML Schema files) required to validate this document as a valid XML document.

The second group of tests is related to the manifest syntactic and semantic validation according to the IMS CP specification. Syntactic validation checks the manifest as an XML file against the IMS CP XML Schema. However, there are other specification restrictions that cannot be expressed by using XML Schema. In e-LD, this kind of restriction is called a *semantic* restriction. An example of semantic restriction might be checking the suitability of references within elements of the manifest. In addition, e-LD includes other semantic tests, such as checking, if all files included inside the .zip file are referenced by an element of the manifest.

The third group of tests is related to the syntactic and semantic validation of the manifest file as an IMS LD UoL. An IMS LD UoL is embedded inside the manifest, so the manifest needs to be validated again against the IMS LD specification control files. Again, another semantic validation is needed, since it is not managed by syntactic (XML-schema oriented) validation. For example, resources tagged

as *imsldcontent* need to be validated to check: 1) if they actually contain IMS LD global elements and 2) if the properties referenced actually exist in the UoL.

All test results are logged in the *importation report* (see Fig. 10). This importation report can be seen as an audit trail that includes three different types of messages: *information*, *warning*, and *error* messages. Information messages are just useful information collected during the process (e.g., how many files have been unzipped). Warning messages inform about problems to be solved (e.g., a package without IMS CP XML schema files). Finally, error messages inform about fatal errors that stop the importation process.

After successful validation, the UoL is ready to be translated into the authoring EML. This translation process is done semiautomatically. The first stage of the translation automatically translates as much as possible from the original UoL into the imported UoL. This translation process is performed in several steps using two guides: 1) the learning flow codified in the source UoL and 2) the IMS LD specification levels. Due to the flow-oriented nature of the authoring EML, the translation process' goal is to extract the implicit learning flow defined in the source UoL. In addition, the learning flow in an IMS LD UoL is defined incrementally at the different IMS LD levels (i.e., the learning flow defined using the elements of level A is enriched with the elements of level B and in the same way level B with elements of level C, although in its current state e-LD does not deal with level C features).

Currently, this translation process automatically translates all elements of IMS Level A and partially translates some Level B elements. The Level B elements addressed are:

- All the additions to the static model, both new elements and modified elements.
- Additions to the completion conditions of the elements in the dynamic model.

The remaining elements are Level B's *conditions*. In IMS LD, conditions are not directly linked to the learning flow, but appear as a separate set of reactive rules, which are also available for all the activities. This feature makes the IMS LD level B very difficult to use and maintain. Even computer science experts without advanced knowledge of rule-based computation systems (a computation model often used in contexts such as artificial intelligence programming [25]) have difficulties using it [26]. Therefore, the direct use of the IMS LD condition may be too complex for the average instructor.

Aside from authoring complexities, there is no algorithm for direct translation from rule-based to flow-based conditions while still preserving the underlying learning design. However, some heuristics can be established and, in each situation, a set of possible translations can automatically be offered.

In addition to the *problems* report incrementally generated during the *Validation* and *Translation* tasks, a new document is generated, the *manual importation report*. The manual importation report records the elements of the source UoL that require manual translation, but also includes some suggestions that can be taken into account for this manual translation process. The result of the *Validation* and *Translation* tasks are: an authoring EML UoL and the validation-importation report are obtained. Both elements are the input for the *Refinement* task.

The *Refinement* task is led by instructors, but with developers' help (particularly with conditions) if needed. Following the developers' suggestions, instructors can modify the UoL to include the original conditions' intended behavior. Additionally, instructors can include small refinements according to their own educational needs.

To effectively assess instructors, developers need, in turn, to understand the UoL. However, UoL understanding requires identifying the relationships of the different elements from the XML document that represent this UoL, which is a very difficult problem, even for skilled developers.

Therefore, e-LD Author provides a browser to help users navigate through an IMS LD UoL. This browser allows for the navigation through different facets of the UoL. This browser provides not only an abstract overview with basic information of the UoL elements but also gives a detailed view where the XML fragment regarding a user-selected IMS LD element is printed for easy review (including hyperlinks to the other referenced elements included in the UoL). These hyperlinks save users' time by avoiding having to search for an element directly in the XML document.

Finally, this browser also includes a dependency viewer that helps users to understand the different relationships between the UoL elements. This dependency viewer provides a powerful tool for the understanding of the learning flow defined in the UoL. This dependency viewer presents an interactive graphical representation of the UoL's *dependency graph*. This dependency graph is a direct graph representing all the relationships between the elements of the UoL related to the sequencing of activities. This graph is built from the XML definition of the UoL, representing the elements of the UoL as nodes and their relationships as edges of the graph. The user can interact with the graph viewer to select which detail of information needs to be displayed. The graph viewer also allows to focus on a particular element.

The different node types included in the dependency graph are the following:

- **Element nodes:** These nodes represent the *plays*, *acts*, *role-parts*, *activity structures*, *learning activities*, and *support activities*.
- **IMSLD content:** They represent learning contents which are a source of property modification that can fire a rule modifying the learning flow.
- **Conditions' rules:** Each condition produces three nodes: one representing the condition as a whole, another one associated to the if part, and a third one associated with the else part.
- **Properties:** Both properties and property groups are represented.

The different edge types included in the dependency graph are as follows:

- **Inclusion dependency:** Edges of this type represent the inclusion relationship between elements. Although all edges are generated, by default, only

inclusion dependencies between the rules' components and between property groups and their components are displayed.

- **Completion dependency:** These edges represent the dependencies between elements and properties regarding the completion of such elements.
- **Visibility modification dependency**: Edges of this type represent visibility modifications produced by fired rules.
- **Property modification dependency:** Edges of this type represent the modification of a property done by: 1) a rule execution, 2) a resource of type *imsldcontent*, or 3) as a result of an element's completion.
- **Property reading dependency:** An edge of this type represents the act of reading the value of a property. The reading of a property's value can be done with multiple purposes: to show some evaluation results in a global element as a completion mechanism for sequencing elements or as part of a condition guard of a condition's rule.

Once the dependency graph is built, properties and conditions are classified according to the classification proposed in Section 2.2. This information is available in the manual import report but it is also attached to the rule node in the dependency graph, so it is accessible from the dependency viewer.

Due to the fact that the dependency graph contains a lot of dependencies, its graphical representation may become unmanageable in a complex UoL, and thus, useless for users. To address that issue the dependency viewer includes a set of configurable filters so that users can control the display of subgraphs.

The next section provides an example of dependency graph and some examples of the analysis that can be performed from the dependency graph.

5.3 Dependency Graph for the Geo-Quiz 3 UoL

Fig. 11 depicts an excerpt of the dependency graph for the Geo-Quiz 3 UoL introduced in Section 4.5 to illustrate which kind of conclusions can be drawn. For example, the dependency graph shows two *imsldcontent* resources (Figs. 11f and 11g) with a different purpose. The *imsldcontent res-questions* (Fig. 11g) have outgoing dependencies to the properties *Answer1, Answer 2, Answer 3, Answer4* (not shown in Fig. 11), and *Answer 5*, so this resource is mainly used to modify properties. In fact, this resource contains the quiz that is proposed to the learner. The modification of these properties fires some rules that modify the properties *Value1, Value2, Value3, Value4* (not shown in Fig. 11), and *Value5* (Figs. 11b, 11c, 11d, and 11e). These properties are later read by the *imsldcontent* resource *res-feedback* (Fig. 11f).

So the overall behavior is that once the test's answers are collected in *res-questions*, some conditions are fired, both preparing feedback (i.e., value properties) and modifying the learning flow accordingly the score (the accuracy property).

Finally, Fig. 11a shows a common pattern where a set of rules modifies the visibility of the same activities (in this example, *flow1*, *flow2*, and *flow3*) but in a complementary way, showing one of them and hiding the others, so that

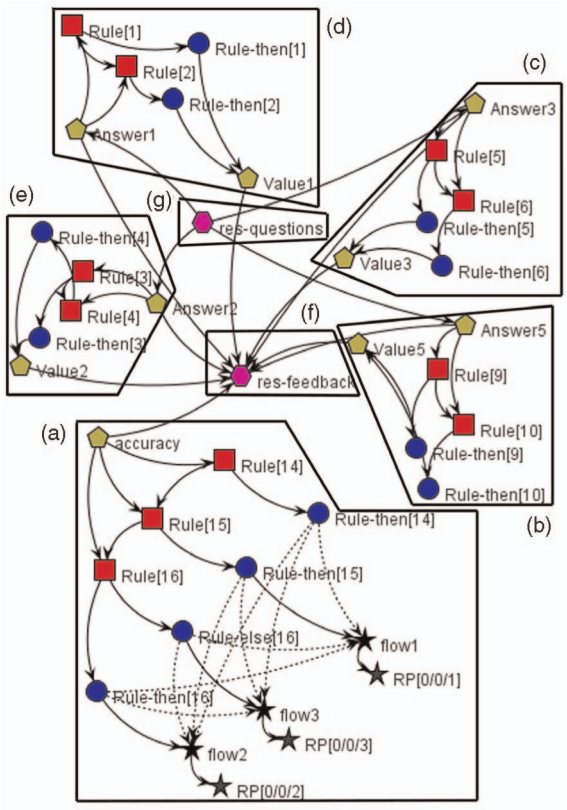

Fig. 11. Dependency graph excerpt of the Geo-Quiz 3 UoL. Stars represent sequencing elements. Hexagons represent *imsldcontent* resources. Pentagons represent properties. Squares represent rules and circles represent rules' parts. Arrows represent dependencies between nodes. Dotted arrows represent a visibility modification dependency, where the element is hidden.

these rules control three conditional learning paths. This is an example of a pattern that can be used as a heuristic during the importation process. The graphical representation of the dependency graph, in which all the dependencies are glued together, facilitates the realization of this kind of analysis, and therefore, the comprehension of the underlying learning design.

6 Related Work

In this section, we present some work related to our authoring and reengineering approach of IMS LD UoLs. As an exhaustive analysis is not possible, we have focused on the different topics and domains that have influenced our approach: 1) business process modeling, 2) instructional design and graphical authoring, and 3) specific IMS LD authoring tools.

The business process management domain is facing a similar problem to ours with the Business Process Management Notation (BPMN) and Business Process Execution Language (BPEL) [20]. BPMN addresses business management from a business analyst's point of view, providing a high-level graphical notation, whereas BPEL provides low-level Web services coordination language that can be used to automate business processes. In fact, a mismatch between BPMN and BPEL has been identified [27]. However, there are proposals to translate BPMN diagrams (with some

restrictions) to the BPEL process [28]. e-LD Author provides high-level notation for the authoring of UoL and then translates it to IMS LD. However, to simplify this translation, the authoring EML retains part of the structure of the exchange EML.

Graphical authoring of EML designs has attracted significant attention in e-learning. There are several EML proposals with built-in graphical notation, such as E2ML [29], PoEML [30], and LAMS [7]. Some of these languages have been proposed just as a design language like E2ML, whereas others cover not only design but also execution aspects like LAMS. A more complete analysis of instructional design languages and graphical notations can be found in [31].

IMS LD authoring is a very popular research topic with a great deal of ongoing work currently being done. Griffiths et al. [32] provide an analysis of the tools needed to work with IMS LD and the difficulties that teachers encounter during the authoring of IMS LD UoLs. Some of the available initiatives are: MOT + [33], ASK-LDT [34], Reload LD Editor [35], CoSMos [36], Prolix GLM [12], ReCourse [37]. In parallel to our work, the last two tools are making a great effort to provide IMS LD with a user-friendly graphical notation.

7 Conclusions and Future Work

In this paper, we have presented our approach, called e-LD, to simplify the authoring and reengineering of IMS LD UoLs. e-LD provides with an EML visual notation system specifically oriented to simplifying UoL authoring. However, it is important to note that we are not promoting yet another new EML. The key idea is to use this author-oriented notation for the authoring process and then export the designs to an EML standard notation (e.g., IMS LD). Our approach also includes a collaborative semiautomatic importation process to address the reuse of preexisting UoLs. The authoring and reengineering processes are supported by the e-LD Author tool.

We have conducted an initial evaluation of the graphical notation included in e-LD and the reengineering process described above has been tested in two different scenarios: with advanced users (e-learning researchers) and PhD students of an e-learning course offered at the Complutense University. As a result of the preliminary experiments, we obtained some evidence on how the graphical notation proposed in e-LD simplifies the process of the authoring UoLs. The creation of UoLs with the complexity of those analyzed in our practical experiments is far beyond the reach of most designers, but students without previous experience in using EMLs were able to recreate them using e-LD. However, further study is required, especially in reuse scenarios, where nonexperts create new UoLs based on UoL templates or previously built UoLs.

Readers who are familiar with UML graphical notation will notice several parallels with the notation described in this work. This is intentional, given the background in Computer Science of the participants in the test case. However, due to the e-LD authoring tool work with the metamodel behind the graphical notation, different notation systems can be developed or customized for a particular stakeholder or community. Therefore, our main short-term goal is to improve the usability of the e-LD Author tool and test our approach with users without a computer science background.

As future work, we will explore new mechanisms to improve the automatic importation of UoLs, particularly conditions' rule clustering and automatic detection of flow-oriented structures. In addition, we will also explore the compatibility with other exchange EMLs like SCORM Sequencing and Navigation [18] and LAMS and whether it is possible to integrate the authoring of UoLs using different target EMLs at design time. We are in early contacts with GRAILS [9], [10] developers to provide an integrated approach that covers from the authoring to publication of the learning designs.

Acknowledgments

The Spanish Committee of Science and Technology (projects TIN2005-08788-C04-01, Flexo-TSI-020301-2008-19, and TIN2007-68125-C02-01) has partially supported this work, as well as the Complutense University of Madrid (research group 921340, Santander/UCM Project PR34/07-15865), and the EU Alfa project CID (II-0511-A).

References

[1] P.B. Sloep, "Reuse, Portability and Interoperability of Learning Content: Or Why an Educational Modeling Language," Online Education Using Learning Objects, R. McGreal, ed., pp. 128-137, Routledge/Falmer, 2004.

[2] IMS: IMS Learning Design Information Model Version 1.0, http://www.imsglobal.org/learningdesign/, 2003.

[3] *Learning Design—A Handbook on Modeling and Delivering Networked Education and Training,* R. Koper and C. Tattersall, eds. Springer Verlag, 2005.

[4] P. Polsani, "Use and Abuse of Reusable Learning Objects," *J. Digital Information,* vol. 3, no. 4, p. 164, 2003.

[5] I. Martínez-Ortiz, J.L. Sierra, and B. Fernández-Manjón, "Enhancing Reusability of IMS LD Units of Learning: The e-LD Approach," *Proc. Eighth IEEE Int'l Conf. Advanced Learning Technologies (ICALT '08),* pp. 402-404, 2008.

[6] I. Martínez-Ortiz, J.L. Sierra, B. Fernández-Manjón, and A. Fernández-Valmayor, "Language Engineering Techniques for the Development of e-Learning Applications," *J. Network and Computer Applications,* to appear.

[7] J. Dalziel, "Implementing Learning Design. The Learning Activity Management System (LAMS)," *Proc. 20th Ann. Conf. Australasian Soc. for Computers in Learning in Tertiary Education (ASCILITE '03),* 2003.

[8] M. Weller, A. Little, P. McAndrew, and W. Woods, "Learning Design, Generic Service Descriptions and Universal Acid," *Educational Technology and Soc.,* vol. 9, no. 1, pp. 138-145, http://www.ifets.info/issues.php?id=30, 2006.

[9] P.J. Muñoz Merino, C. Delgado Kloos, and J. Fernández Naranjo, "Enabling Interoperability for LMS Educational Services," *Computer Standards & Interfaces,* vol. 31, no. 2, pp. 484-498, 2009.

[10] R. Hernández, A. Pardo, and C. Delgado Kloos, "Creating and Deploying Effective eLearning Experiences Using .LRN," *IEEE Trans. Education,* vol. 50, no. 4, pp. 345-351, Nov. 2007.

[11] R. Koper and D. Burgos, "Developing Advanced Units of Learning Using IMS Learning Design Level B," *Int'l J. Advanced Technology for Learning,* vol. 2, no. 4, pp. 252-259, 2005.

[12] S. Heyer, P. Oberhuemer, S. Zander, and P. Prenner, "Making Sense of IMS Learning Design Level B: From Specification to Intuitive Modeling Software," *Proc. Second European Conf. Technology Enhanced Learning (EC-TEL '07),* pp. 86-100, 2007.

[13] W. Dick and L. Carey, *The Systematic Design of Instruction,* fourth ed. Harper Collins College Publishers, 1996.

[14] P. Sloep, H. Hummel, and J. Manderveld, "The Learning Design Specification," *Learning Design—A Handbook on Modeling and Delivering Networked Education and Training,* R. Koper and C. Tattersall, eds., pp. 139-160, Springer, 2005.

[15] R.S. Aguilar-Saven, "Business Process Modeling: Review and Framework," *Int'l J. Production Economics,* vol. 90, no. 2, pp. 129-149, 2004.
[16] F. Ferrucci, G. Tortora, and G. Vitiello, "Exploiting Visual Languages in Software Engineering," *Handbook of Software Engineering and Knowledge Engineering,* pp. 53-76, 2002.
[17] *IMS: IMS Content Packaging Information Model Version 1.1.4,* http://www.imsglobal.org/content/packaging/cpv1p1p4/imscp_infov1p1p4.html, Dec. 2008.
[18] ADL, Shareable Content Object Reference Model (SCORM), third ed., Sequencing and Navigation Version 1.0, http://www.adlnet.gov/scorm/, 2006.
[19] P.P. Chen, "Database Design Using Entities and Relationships," *Principles of Data Base Design,* S.B. Yao, ed., pp. 174-210, Prentice-Hall, 1985.
[20] W. Aalst and H. Kees, *Workflow Management: Models, Methods, and Systems.* MIT Press, 2004.
[21] I. Martínez-Ortiz, P. Moreno-Ger, J.L. Sierra-Rodríguez, and B. Fernández-Manjón, "Supporting Authoring and Operationalization of Educational Modeling Languages," *J. Universal Computer Science,* vol. 13, no. 7, pp. 938-947, 2007.
[22] I. Martínez-Ortiz, J.L. Sierra, and B. Fernández-Manjón, "Translating E-Learning Flow-Oriented Activity Sequencing Descriptions into Rule-Based Designs," *Proc. Sixth Int'l Conf. Information Technology: New Generations (ITNG '09),* to appear.
[23] P. Laforcade, "Graphical Representation of Abstract Learning Scenarios: The UML4LD Experimentation," *Proc. Seventh IEEE Int'l Conf. Advanced Learning Technologies (ICALT '07),* pp. 477-479, 2007.
[24] D. Burgos, "Geo-Quiz 3," http://hdl.handle.net/1820/404, Nov. 2008.
[25] L. Brownston, R. Farell, E. Kant, and N. Martin, *Programming Experts Systems in OPS5: An Introduction to Rule-Based Programming.* Adisson-Wesley, 1985.
[26] X. Li, "What's So Bad About Rule-Based Programming?" *IEEE Software,* vol. 8, no. 5, pp. 103-105, Sept. 1991.
[27] J. Recker and J. Mendling, "On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages," *Proc. 18th Int'l Conf. Advanced Information Systems Eng.,* pp. 521-532, 2006.
[28] C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede, "Translating BPMN to BPEL," BPM Center Report BPM-06-02, http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2006/BPM-06-02.pdf, 2006.
[29] L. Botturi, "E2ML: A Visual Language for the Design of Instruction," *Educational Technology Research and Development,* vol. 54, no. 3, pp. 265-293, 2006.
[30] M. Caeiro, M.J. Marcelino, M. Llamas, L. Anido-Rifón, and A.J. Mendes, "Supporting the Modeling of Flexible Educational Units PoEML: A Separation of Concerns Approach," *J. Universal Computer Science,* vol. 13, no. 7, pp. 980-990, 2007.
[31] L. Botturi, M. Derntl, E. Boot, and K. Figl, "A Classification Framework for Educational Modeling Languages in Instructional Design," *Proc. Sixth IEEE Int'l Conf. Advanced Learning Technologies (ICALT '06),* pp. 1216-1220, 2006.
[32] D. Griffiths, J. Blat, R. Garcia, H. Vogten, and K.L. Kwong, "Learning Design Tools," *Learning Design—A Handbook on Modeling and Delivering Networked Education and Training,* R. Koper and C. Tattersall, eds., pp. 109-135, Springer, 2005.
[33] G. Paquette, M. Léonard, K. Lundgren-Cayrol, S. Mihaila, and D. Gareau, "Learning Design Based on Graphical Knowledge-Modeling," *Educational Technology & Soc.,* vol. 9, no. 1, pp. 97-112, 2006.
[34] P. Karampiperis and D. Sampson, "A Flexible Authoring Tool Supporting Adaptive Learning Activities," *Proc. Int'l Assoc. for Development of the Information Soc. (IADIS) Int'l Conf. Cognition and Exploratory Learning in Digital Age (CELDA '04),* 2004.
[35] C.D. Milligan, P. Beauvoir, and P. Sharples, "The Reload Learning Design Tools," *J. Interactive Media in Education,* vol. 2005, no. 7, 2005.
[36] Y. Miao, "Facilitating Learning Designers to Author Units of Learning Using IMS LD," *Proc. Int'l Conf. Computers in Education,* pp. 275-282, 2005.
[37] D. Griffiths, P. Beauvoir, and P. Sharples, "Advances in Editors for IMS LD in the TENCompetence Project," *Proc. Eighth IEEE Int'l Conf. Advanced Learning Technologies (ICALT '08),* pp. 1045-1047, 2008.

Iván Martínez-Ortiz received the MS degree in computer science from the Complutense University of Madrid (UCM) in 2004. He is currently working toward the PhD degree at UCM, where he is a member of the <e-UCM> Group, the research group on e-learning technologies of the university. From 2004 to 2007, he was a lecturer at Centro de Estudios Felipe II, Aranjuez, Spain. He has been a full-time lecturer of computer science in the Computer Science School at UCM since 2007. He has coauthored more than 30 research papers published in international journals and international conferences. His research interests include e-learning technologies, authoring in e-learning, and the integration of educational modeling languages and workflows technologies.

José-Luis Sierra received the BS and MS degrees from the Technical University of Madrid (UPM) in 1991, 1992, and 1995, and the PhD degree in computer science from the Complutense University of Madrid (UCM) in 2004. In 1998, he joined UCM as an assistant professor, where he has been an associate professor of computer science since 2007. He was a research assistant in the Industrial Automatic Institute at the Spanish Scientific Research Council during 1990-1992 and the Artificial Intelligence Department at UPM during 1993-1998. He has coauthored more than 70 research papers published in international journals and international conferences. His research interests include e-learning technologies, domain-specific languages, and markup languages. He is a member of the <e-UCM> Research Group.

Baltasar Fernández-Manjón received the MS and PhD degrees in physics from the Complutense University of Madrid (UCM) in 1989 and 1996. In 1992, he joined UCM as an assistant professor of computer science, where he has been an associate professor since 1998. Currently, he is the vice dean of research and foreign relationships in the Computer Science School at UCM. He has coauthored more than 90 research papers published in international journals and international conferences. His main research interests are e-learning technologies, educational uses of markup technologies, application of educational standards, and user modeling. He leads the <e-UCM> Research Group. He is a senior member of the IEEE.