A Learning Design Ontology based on the IMS Specification

Ricardo R. Amorim, Manuel Lama, Eduardo Sánchez, Adolfo Riera and Xosé A. Vila
Department of Electronics and Computer Science, Faculty of Physics
Campus Univesitario Sur s/n, University of Santiago de Compostela
15782 Santiago de Compostela, A Coruña, Spain
rramorin@usc.es
lama@dec.usc.es
eduardos@usc.es
eladolfo@usc.es
vila@dec.usc.es

ABSTRACT

In this paper, we present an ontology to represent the semantics of the IMS Learning Design (IMS LD) specification, a meta-language used to describe the main elements of the learning design process. The motivation of this work relies on the expressiveness limitations found on the current XML-Schema implementation of the IMS LD conceptual model. To solve these limitations, we have developed an ontology using Protégé at the knowledge level. In addition, we provide its implementation in OWL, the standard language of the Semantic Web, and the set of associated axioms in first-order logic. The OWL file is available at *http://www.eume.net/ontology/imsld_a.owl*.

Keywords

IMS Learning Design, Ontologies, Semantic description, Formal axioms

Introduction

In the last years, the popularity of the Internet has opened the door to new ways of learning and numerous educational tools and applications. In this context, the need to manage reusable resources has driven the development of several metadata specifications in order to represent learning content, educational resources and learning design methodologies. This paper focuses on the representational issues of the *learning design*, which describes the method that enables learners to achieve learning objectives after carrying out a set of activities using the resources of an environment.

The specifications for the learning design, known as Educational Modelling Languages (EML), are models of semantic information and aggregation that describe, from a pedagogic point of view, the content as well as the educational activities. These elements are organized into units of study with the aim of allowing their reuse and interoperability (Rawlings et al., 2002). Moreover, EMLs facilitate the description of pedagogic aspects that are related with LOs in educational processes (Koper, 2001). The principal EML specifications are as follows:

- *CDF* (http://www.ariadne-eu.org). It uses the ARIADNE Course Description Format (A-CDF) for the description of courses (Verbert & Duval, 2004). A course in A-CDF consists of XML documents along with a course generator LMS. It places special emphasis on the content and its aggregation, but it is expressive enough to describe the learning process in accordance with a pedagogic model. The didactic material that can be managed through CDF is restricted to text format. It uses a combination of tools developed by the ARIADNE consortium (curriculum editors, LMS, KPS) and establishes the concept of *Course* as a unit of study.
- *LMML* (http://www.lmml.de). An acronym of Learning Material Mark-up Language. It is based on a meta-model in order to be used in different application domains. LMML relies on XML for the description of e-learning material (Slavin et al., 1995), and comprises various learning material modules, each one containing other sub-modules. Focused on a conceptual, modular and hierarchical structure of e-learning content, LMML can be adapted to different learning situations and students. It uses the concept of *Course* as a unit of study.
- *PALO* (http://sensei.lsi.uned.es). It is a modelling language that has been developed by the UNED (Universidad Nacional de Enseñanza a Distancia, Spain) (Rodríguez-Artacho et al., 1999). PALO describes courses organized into modules that contain learning activities, content, and an associated teaching plan. The language provides templates to define types of Learning Scenarios with their associated pedagogic properties. By using the language features, it is possible to establish the sequencing of both modules and learning tasks. According to the course constraints, these attributes also allow to define deadlines and dependencies between modules and tasks. It uses the concept of *Module* as a unit of study.

- *Targeteam* (http://www.targeteam.net). An acronym of TArgeted Reuse and GEneration of TEAching Materials. This language supports the production and management (use and reuse) of learning material (Koch, 2002), including notes and contents such as explanations, motivation, and examples. All these elements can be carefully structured in an interrelated manner. It is focused on the use of an XML-based language, TeachML, and uses the concept of *Issue* as a unit of study. This EML allows the use of material in different learning situations and pedagogic domains (primary, secondary and higher education).
- *TML/Netquest* (http://www.ilrt.bris.ac.uk/netquest). It uses the Tutorial Mark-up Language (TML), an extension of HTML, to produce questions (Brickley, 1996). This language was designed to separate the semantic content of the layout, or on-screen format, from a question. The TML files are in text format, and can be generated from other formats or other questions in a database. This EML does not support the concept of a unit of study.
- *IMS Learning Design (IMS LD)* (http://www.imsglobal.org/learningdesign). This specification, drawn up by the IMS/LDWG work group, is an integration of the EML developed by the OUNL (Open University of Netherlands), with other existing IMS specifications for the exchange and interoperability of e-learning material. The OUNL EML is a meta-vocabulary that is defined based on the diversity of concepts existing in a wide range of pedagogic techniques. The IMS EML incorporates the OUNL EML, and describes the structure and educational processes based on a pedagogic metamodel, using units of learning called *Learning Design* (IMS, 2003a). IMS LD describes a method that is made up of a number of activities carried out by both learner and staff in order to achieve some learning objectives. It allows the combination of various techniques (traditional, collaborative, etc.), and facilitates the description of new ones. From the proposed specifications, the IMS LD has emerged as the *de facto* standard for the representation of any learning design that can be based on a wide range of pedagogical techniques.

The metadata specifications are useful to describe educational resources, and thus to facilitate interoperability and reuse between learning software platforms, as they represent the vocabulary describing the different aspects of the learning process. However, the main drawback is that the meaning of the specification is usually expressed in *natural language*. Although this description is easy to understand for humans, it would be difficult to be automatically processed by software programs. To solve this issue, ontologies (Gómez-Pérez et al., 2004) come handy to describe *formally* and *explicitly* the structure and meaning of the metadata elements; that is, an ontology would semantically describe the metadata concepts. In the educational domain, several ontologies have been proposed: (1) to describe the learning contents of technical documents (Kabel et al., 1999); (2) to model the elements required for the design, analysis, and evaluation of the interaction between learners in computer supported cooperative learning (Inaba et al., 2001); (3) to specify the knowledge needed to define new collaborative learning scenarios (Barros et al., 2002); or (4) to formalize the semantics of learning objects that are based on metadata standards (like LOM) (Brase & Nejdl, 2004). The focus of that research is either on the development of a taxonomy of concepts on the basis of a established theory or specification (1 to 3), or on the formal definition of the metadata using an ontology language (4). However, none of them deal with the formal description of the meaning of the concepts, and they do not address the ontological modelling of any specification for learning design.

In this paper, we present a *learning design ontology* based on the IMS LD specification, the *de facto* meta-language for the learning design. In this ontology, the IMS LD elements are modelled in a *concept taxonomy* in which the relations between the concepts are explicitly represented. Furthermore, a set of *axioms* constraining the semantics of the concepts has been formulated from the restrictions (expressed in natural language) identified in the analysis of the IMD LD specification.

The paper is structured as follows: in the next section, the limitations of the XML-Schema language on representing the IMS LD specification are outlined; then, the concept taxonomy and the ontology axioms are described, and an example illustrates how the ontology could be used; then, a description of how the ontology was implemented and used in an educational environment is presented; and finally, the presented work is discussed and the main contributions are summarized.

The Need for a Learning Design Ontology

The IMS Learning Design specification is a meta-language that describes all the elements of the design of a teaching-learning process (IMS, 2003a). This specification is based on: (1) a well-founded conceptual model that defines the vocabulary and the functional relations between the concepts of the LD; (2) an information model that describes in an informal (natural language) way the semantics of every concept and relation introduced in the conceptual model; and (3) a behavioural model that specifies the constraints imposed to the software system

when a given LD is executed in runtime. In other words, the behavioural model defines the semantics of the IMS LD specification during the execution phase.

To facilitate the interoperability between software systems, the IMS LD specification has been formally modelled through the XML-Schema language (Thompson et al., 2004; IMS, 2003b). However, the knowledge model of this language is not expressive enough to describe the semantics (or meaning) associated to the elements of the IMS LD. Thus, the main limitations of the XML-Schema language are (Gil & Ratnakar, 2002):

Hierarchical (is-a) relations between two or more concepts cannot be explicitly defined. Therefore, there are no inheritance mechanisms facilitating the representation of concept taxonomies. For example, in the IMS LD specification, the Learner and Staff elements do not inherit the attributes and relations of the Role element: they are just included as XML sub-elements of the Role element. Figure 1 compares both the OWL and XML-Schema specifications based on the definition of the Learner concept. In OWL, the hierarchical relation between the Learner and Role concepts is explicitly defined through the rdfs:subClassOf, and, therefore, the attributes of Role will be automatically inherited by Learner. In XML-Schema, however, the attributes of Role are directly added to the definition of Learner, but no hierarchical relation is established between them.

- Properties of relations cannot be defined. XML-Schema language does not provide primitives to represent neither mathematical properties (like symmetry or transitivity) nor taxonomic properties (like disjoint and exhaustive partitions) of a relation. For example, the IMS LD specifies that an instance of the Staff cannot be a Learner for any given unit of learning, what means that Staff and Learner are disjoint concepts. Figure 1 shows how this taxonomic property can be explicitly expressed in OWL.
- General and formal constraints (or axioms) between concepts, attributes, and relations cannot be specified. These axioms describe more precisely the semantics of the concepts as they constrain how the instances of the concepts could be created. For instance, the axiom "if an Act is executed in the context of a Play, and both have a given value for the time limit attribute, the value of this attribute for the Play should be greater or equal than the value for the Act" could not be represented in the XML-Schema language. This restriction, however, could be defined in ontology languages like F-Logic (Kiefer et al., 1995) or Ontolingua (Gruber, 1993). In order to describe these kinds of axioms in OWL, a new language, called SWRL (Horrocks et al., 2004), has been submitted to the W3C.

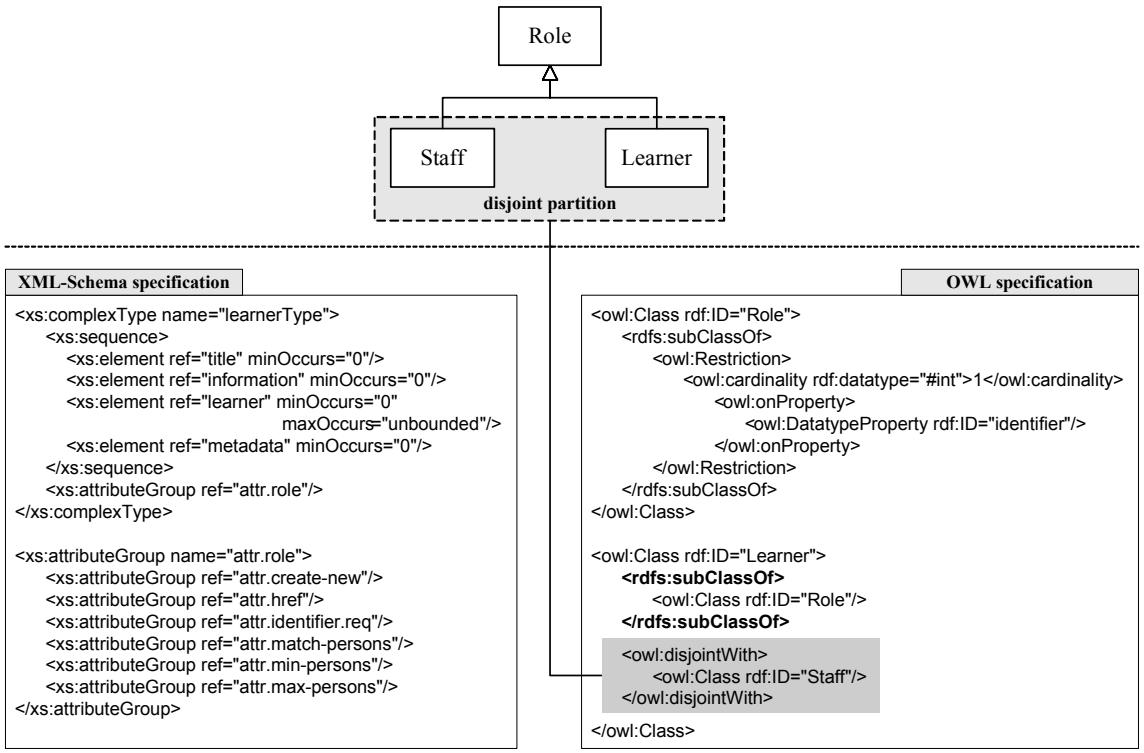


Figure 1. Comparison between the OWL and XML-Schema specifications

Therefore, the IMS LD specification requires a modelling capable of describing *explicitly and formally* the semantics of its elements. To achieve this goal, we have developed an ontology (Gómez-Pérez et al., 2004), which facilitates the semantic description of the conceptual model as well as the definition of formal axioms related to both information and behavioural models. This ontology is based on a knowledge model that includes

complex taxonomic relations (like both hierarchical and ad-hoc relations, disjoint and exhaustive partitions, etc.) as well as formal axiom descriptions.

The Learning Design Ontology

To develop the Learning Design ontology we have created a *concept taxonomy*, which describes the elements of the IMS LD conceptual model and the IMS LD information model, and a *set of axioms*, which formally constraint the semantics of the concept taxonomy on the basis of the explanations formulated in natural language in both information and behavioural models.

Description of the Concept Taxonomy

The upper node of the LD ontology is the Unit of Learning concept (Figure 2) that defines a general module of an educational process, like a course or a lesson. Following the IMS LD specification, a unit of learning is modelled as a content package (IMS, 2003a) that integrates the description of both the LD and the set of resources related to it. The Resource concept allows representing various entities, like physical resources (Web pages, files, etc.), and concepts whose attribute description is domain-dependent (learning objectives, prerequisites, etc.). To model the different kinds of resources, we have extended the IMS LD specification with a new hierarchy of concepts (grey boxes in Figure 2). In this way, when a LD concept refers to any of the resource properties, it establishes a relation with the Item concept, which in turn, has a set of subclasses that replicate the hierarchical structure of the resources (following a one-to-one correspondence). These two hierarchies have been introduced to decouple the references to the resources (Item hierarchy) from their modelling (Resource hierarchy). Thus, if two applications use the same LD to model a course, but define resources in a different way (for example, if the learning objectives are specified either as textual description or through their corresponding attributes), the LD does not need to be changed because the links to the resources are indirectly established through the Item hierarchy.

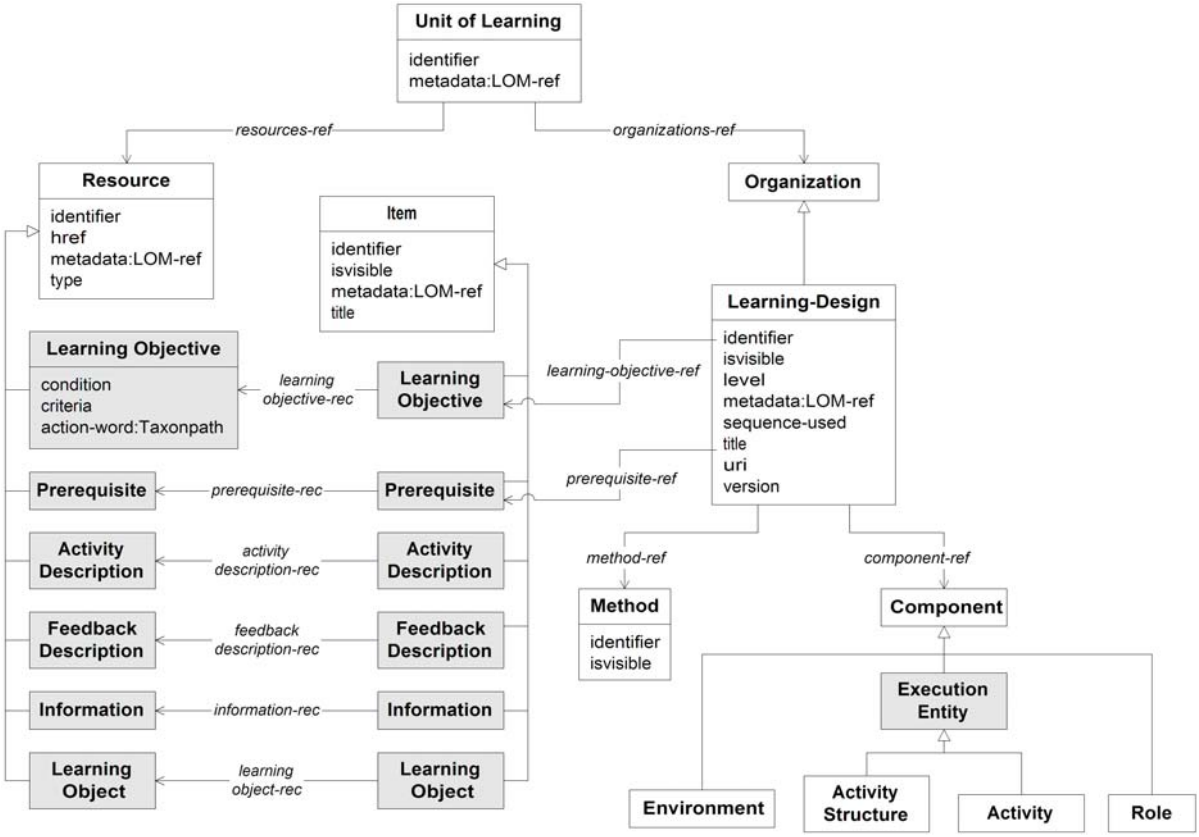


Figure 2. Upper concepts of the Learning Design ontology

Learning Design Description

The Learning Design concept is related to the Learning Objective and Prerequisite concepts, which define the intended outcomes when the unit of learning is carried out, and the previous knowledge needed to participate in it, respectively. Both concepts are subclasses of the Item concept, and therefore they will be mapped onto the Learning objective and Prerequisite concepts of the Resource hierarchy.

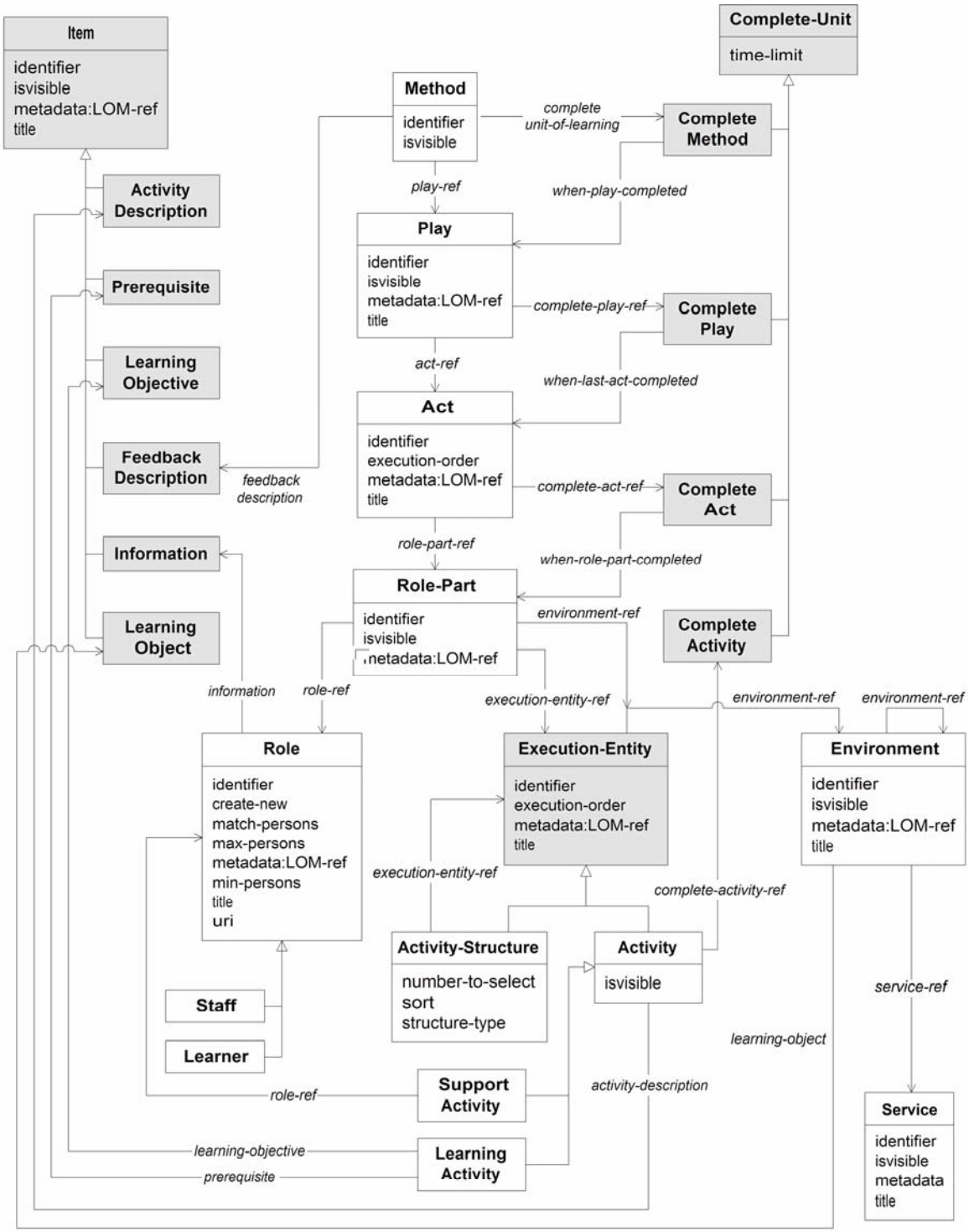


Figure 3. Concept taxonomy that describes the dynamics of a learning design

The Learning Design concept has a number of Components used to describe the learning process: the Execution Entities to be carried out, which can be Activities or Activity Structures (groups of activities that will be executed in sequence); the Roles that participate in the execution of those activities as instances of the Learner

and Staff concepts; and the Environments that describe the educational resources to be used in the activities. These concepts constitute an exhaustive and disjoint partition, because an instance of a Component must necessarily be an instance of one of its subclasses.

The Learning Design concept is also related to the Method concept, which describes the dynamics of the learning process (Figure 3): a method is composed of a number of instances of the Play concept that could be interpreted as the runscript for the execution of the unit of learning. All the play instances have to be executed in parallel, and each one consists of Act instances, which could be understood as a stage of a course or module. The Act instances must be executed in sequence (according to the values of the execution order attribute), and they are composed of a number of Role Part instances that will be executed concurrently. A Role Part associates a Role(s) with an Execution Entity to be carried out in the context of the act. Finally, every Execution Entity requires an Environment, which manages Learning Objects as resources. In summary, the execution of an act consists on the simultaneous participation of roles in an activity or group of activities, and once the activities are completed, the associated roles could participate in the execution of any other activity through different role part instances.

The Activity concept has two subclasses: the Learning Activity concept and the Support Activity concept. A Learning Activity models an educational activity that establishes a relation with the Prerequisite and the Learning Objective concepts. The Support Activity, however, is introduced to facilitate the execution of a learning activity, but it does not cover any learning objective. These two classes constitute a disjoint and exhaustive partition, because an instance of the Activity concept should be either a learning or a support activity.

Every concept involved in the dynamics of the learning process (Method, Play, Act, and Activity) establishes a relation with one of the subclasses of the Complete Unit concept, which indicates when an execution is finished. In the IMS LD Level A, this condition can be specified through the time limit attribute, which defines the temporal duration of the execution, or referred to an instance of the entity of which it is composed. For example, an act would be completed when the instance of the Role Part indicated by the relation when-role-part-completed has finished. Furthermore, in both Level B and C of IMS LD, the modelling of these subclasses will be extended to enable the specification of more complex completion conditions.

Description of the Learning Design Ontology Axioms

From a modelling point of view, the formal definition of the semantic constraints of the LD concepts is the main advantages of the learning design ontology when compared with the IMS LD XML-Schema specification. On one hand, the semantics of the concepts are *completely* included in the ontology (not only the taxonomic structure), and, on the other hand, the programmers of LD software systems will not need to understand the descriptions of the IMS LD models in order to translate its meaning in sentences of programming code.

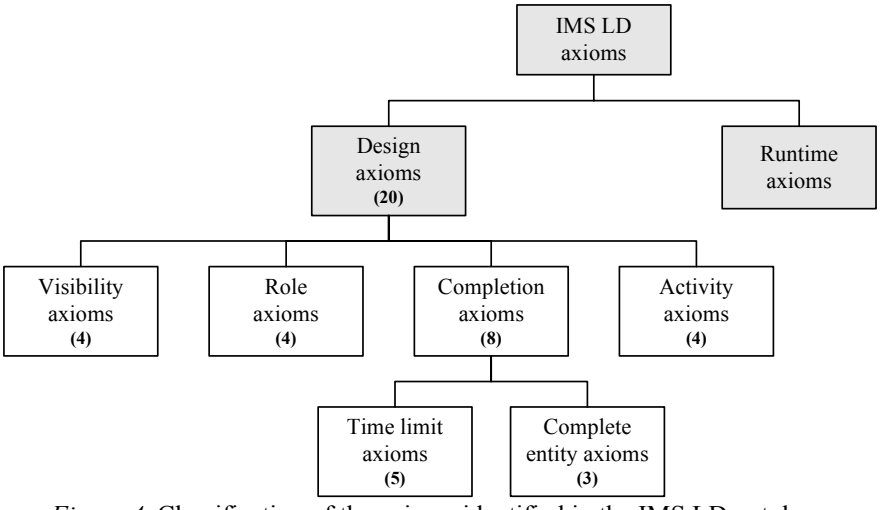


Figure 4. Classification of the axioms identified in the IMS LD ontology

The three models of the IMS LD specification contain a natural language description of the semantics of all the taxonomy concepts, including the *constraints* that should verify their instances when created and managed by a

software system. To incorporate these restrictions to the LD ontology we have applied the following procedure: first, the description of the constraints is identified in the text of IMS LD; then, if necessary, this description is reformulated considering the elements of the LD concept taxonomy (concepts, relations, and attributes); and, finally, the restrictions are represented in a declarative, formal, and language-independent way as *axioms* declared in first order logic. Following this procedure, we have identified and formalized a number of axioms that can be classified depending on the phase where the axioms are applied. Thus, two general kinds of axioms are distinguished (Figure 4): design axioms and runtime axioms.

On one hand, *runtime axioms* are associated with the management and monitoring of the execution of the learning design created during the design phase. For example, one of the axioms of this category should guarantee that the plays that compose a method will be executed in parallel. However, to specify many of these axioms it is necessary to *extend* the LD ontology for including a runtime model (not defined in the IMS LD specification) that would represent the different states of execution of the learning design. Most of these axioms have been extracted from the behavioural model.

On the other hand, *design axioms* determine how the instances of the taxonomy concepts will be created when a given learning design has been specified. For example, the first axiom of Table 2 will not allow the creation of a method with a value for the time limit attribute less than the time limit of any of its plays. This kind of axioms guarantee the consistence of all the components of the design created for modelling a unit of learning. Currently, we have extracted and formally defined 20 axioms, most of them from the IMS LD concept and information models. In this paper, we will focus on the description of this kind of axiom.

Design Axiom Description

The axioms have been specified in first order logic as sentences made up of: an antecedent, which contains the conditions to be verified, and a consequent, which describes the constraints to be applied to the concepts, attributes or relations of the ontology (these ontology elements are the universe of discourse). According to this, the axioms could be classified on the basis of the ontology elements whose values or relations are affected by the axiom's consequent part. The following types have been identified (Figure 4): *completion* axioms, *role* axioms, *visibility* axioms, and *activity-related* axioms.

The completion axioms are obtained from the restrictions related to the ending conditions of the elements involved in the runscript (Method, Play, Act and Role-Part). Particularly, these axioms specify both the values of the attributes and relation range associated to the sub-lasses of the Complete Unit concept. Considering this, two kinds of completion axioms could be distinguished (Figure 4): time limit axioms, and entity completion axioms.

The entity completion axioms (Table 1) will restrict the instances of the concepts associated to the range of the relations when-last-act-completed, when-play-completed, and when-role-part-completed, which indicate what are the runscript elements, REI, whose execution marks the ending of a given runscript element, REG. Taking this into account, this kind of axioms establishes that REI must be necessarily a subset of the runscript elements executed in the context of REG.

Table 1. Formal definition of the entity completion axioms

1	IMS LD Specification	*Page 42 (item 0.4.1)*: "This element states that an act is completed when the referenced role-part(s) is (are) completed. More than one role-part can be selected, meaning that all the referenced role-parts must be completed before the act is completed."
	Explanation	The `Role Part(s)` referred as the value of the attribute `when-role-part-completed` of an `Act` must be a subset of the `Role Part(s)` associated to the `Act`.
	Formal Description	$\forall$ a, ca, rp $\mid$ a $\in$ Act $\wedge$ ca $\in$ Complete-Act $\wedge$ complete-act-ref(ca, a) $\wedge$ rp $\in$ Role-Part $\wedge$ when-role-part-completed(rp, ca) $\rightarrow$ role-part-ref(rp, a)
2	IMS LD Specification	*Page 40 (item 0.5.1)*: "This element states that a play is completed when the last act is completed."
	Explanation	The `Act` referred as the value of the attribute `when-last-act-completed` of a `Play` must be one of the `Acts` associated to the `Play`.
	Formal Description	$\forall$ p, cp, a $\mid$ p $\in$ Play $\wedge$ cp $\in$ Complete-Play $\wedge$ complete-play-ref(cp, p) $\wedge$ a $\in$ Act $\wedge$ when-last-act-completed(a, cp) $\rightarrow$ act-ref(a, p)

3	IMS LD Specification	*Page 38 (item 0.4.1)*: "This element states that an unit-of-learning is completed when the referenced play(s) is (are) completed. More than one play can be selected, meaning that all the referenced plays must be completed before the unit-of-learning is completed."
	Explanation	The `Play(s)` referred as the valued of the attribute `when-role-part-completed` of a `Method` must be a subset of the `Plays` associated to the `Method`.
	Formal Description	$\forall$ m, p, cm $\mid$ m $\in$ Method $\wedge$ p $\in$ Play $\wedge$ cm $\in$ Complete-Method $\wedge$ complete-method-ref(cm, m) $\wedge$ when-play-completed(p, cm) $\rightarrow$ play-ref(p, m)

The time limit axioms (Table 2) constrain the possible values of the attribute time-limit that indicates the time interval in which a runscript element is executed. Following the IMS LD specification, for any runscript element the origin of this interval will be the time instant associated to the beginning of the unit of learning. Taking this into account, this kind of axioms establishes that: if the time-limit attribute of the Complete Unit related with a runscript element REI has assigned a value, it must be necessarily greater than the values of the attribute of the Complete Units related to the runscript elements executed in the context of REI (axioms 4 to 6); and the values of the time-limit attribute for the Complete Unit concept must be consistent with the values of the execution-order and structure-type attributes associated with the Acts and Activity Structures concepts respectively (axioms 7 and 8). These axioms guarantee the correct design of the runscript elements whose execution is in sequence.

Table 2. Formal definition of the time limit axioms

4	IMS LD Specification	*Page 38 (item 0)*: "The method contains a sequence of elements for the definition of the dynamics of the learning process. It consists of one or more play(s)." *Page 38 (item 0.2.2)*: "The time limit specifies that it is completed when a certain amount of time has passed, relative to the start of the run of the current unit of learning. The time is always counted relative to the time when the run of the unit-of-learning has started. Authors have to take care that the time limits of role-parts, acts and plays are logical."
	Explanation	The value of the attribute `time limit` associated to a `Method` (through its `Complete Method`) must be greater than the value of the `time-limit` associated to any `Play` (through its `Complete Play`) that is executed in the context of the `Method`. That is, the `Plays` cannot finish after the `Method`.
	Formal Description	$\forall$ m, p, cm, cp $\mid$ m $\in$ Method $\wedge$ p $\in$ Play $\wedge$ cm $\in$ Complete-Method $\wedge$ cp $\in$ Complete-Play $\wedge$ play-ref(p, m) $\wedge$ complete-unit-of-learning-ref(cm, m) $\wedge$ complete-play-ref(cp, p) $\rightarrow$ time-limit(cm) $\geq$ time-limit(cp)
5	IMS LD Specification	*Page 39 (item 0)*: "A Play consists of a series of acts and an act consists of a series of role-parts. When there is more than one play, these are interpreted concurrently and independent of each other." *Page 40 (item 0.5.2):* As the item 0.2.2 in page 38.
	Explanation	The value of the attribute `time-limit` associated to a `Play` (through its `Complete Play`) must be greater or equal than the value of the `time limit` associated to any `Act` (through its `Complete Act`) that is executed in the context of the `Play`. That is, an `Act` cannot finish after the `Play`.
	Formal Description	$\forall$ p, cp, a, ca $\mid$ p $\in$ Play $\wedge$ cp $\in$ Complete-Play $\wedge$ complete-play-ref(cp, p) $\wedge$ a $\in$ Act $\wedge$ ca $\in$ Complete-Act $\wedge$ complete-act-ref(ca, a) $\wedge$ act-ref(a, p) $\rightarrow$ time-limit(cp) $\geq$ time-limit(ca)
6	IMS LD Specification	*Page 42 (item 0.4.2)*: As the item 0.2.2 in page 38. *Page 41 (item 0.3)*: "A play consists of a series of acts and an act consists of a series of role-parts. A role-part relates exactly one role to exactly one type of activity (including the performance of another unit-of-learning and activity-structures). Role-parts within one act, are performed concurrently."
	Explanation	The value of the attribute `time limit` associated to an `Act` (through its `Complete Act`) must be greater than the value of the `time-limit` associated to any `Activity` related to a `Role-Part` that is executed in the context of the `Act`. That is, the `Role-Parts` cannot finish after the `Act`.
	Formal Description	$\forall$ a, ca, actv, as, rp $\mid$ a $\in$ Act $\wedge$ ca $\in$ Complete-Act $\wedge$ complete-act-ref(ca, a) $\wedge$ rp $\in$ Role-Part $\wedge$ role-part-ref(a, rp) $\wedge$ actv $\in$ Activity $\wedge$ cactv $\in$ Complete-Activity $\wedge$ complete-activity-ref(cactv, actv) $\wedge$ as $\in$ Activity-Structure $\wedge$ (execution-entity-ref(actv, rp) $\vee$ (execution-entity-ref(as, rp) $\wedge$ execution-entity-ref(actv, as))) $\rightarrow$ time-limit(ca) $\geq$ time-limit(cactv)
7	IMS LD Specification	*Page 41 (item 0)*: "When there is more than one act in a play, these are presented in sequence from first act to last act. Only one act in a play is the active act at any moment in time, starting with the first. When the first act is completed, the second act is made the active act. When the second act is completed, the third act is made active, etc."
	Explanation	If the value of the attribute `execution-order` of an `Act` is greater than the value of the `execution-order` for other `Act`, and both `Acts` are executed in the context of a same `Play`, the value of the attribute `time-limit` associated to the first `Act` (through its `Complete Act`) is greater than the value of the attribute for the second `Act`.

	Formal Description	∀ p, a, b, ca, cb | p ∈ Play ∧ a, b ∈ Act ∧ act-ref(a, p) ∧ act-ref(b, p) ∧ ca, cb ∈ Complete-Act ∧ complete-act-ref(ca, a) ∧ complete-act-ref(cb, b) ∧ (execution-order(a) ≥ execution-order(b)) → time-limit(ca) ≥ time-limit(cb)
8	IMS LD Specification	*Page 31 (item 0)*: "An activity structure groups activities in sequences or selections."
	Explanation	If the value of the attribute `structure-type` of an `Activity Structure` is "sequence", and there are two activities executed in the context of the same `Activity Structure` with consecutive values for the attribute `execution order`, the value of the attribute `time limit` associated to these `Activities` must be consistent with the values of the `execution order` attribute.
	Formal Description	∀ as, a, ca, b, cb | as ∈ Activity-Structure ∧ structure-type(as) = "sequence" ∧ a, b ∈ Activity ∧ execution-entity-ref(a, as) ∧ execution-entity-ref(b, as) ∧ complete-activity-ref(ca, a) ∧ complete-activity-ref(cb, b) ∧ (execution-order(a) = execution-order(b) + 1) → time-limit(ca) ≥ time-limit(cb)

The visibility axioms (Table 3) restrict the value of the attribute isvisible associated to the learning design elements, establishing the conditions under which they can be accessible to the user through a graphical interface. Based on that, these axioms determine the value of the attribute isvisible for the following elements: the Activities executed in the context of an Activity Structure (axiom 9); the Environments used in the execution of either Activity Structures or Activities (axioms 10 and 11); and the Prerequisites and Learning Objectives related to the learning design (axiom 12). This kind of axiom must also be applied in the runtime phase as the visibility constraints have to be guaranteed during the execution of the unit of learning.

Table 3. Formal definition of the visibility axioms

9	IMS LD Specification	*Page 83 (item 17)*: "Environments are connected to activities, activity-structures or roles (in a role-part). When an activity-description is visible, always the connected environment (including the content structure of the environment) must be made visible. It must be possible to access and see the activity-description and the content of one of the objects or services within the environment at the same time."
	Explanation	When the value of the `isvisible` attribute of an `Activity Description` is "true", the value of that attribute for the `Environments` connected to the `Activity` associated to the `Activity Description` must be also true.
	Formal Description	∀ a, ad, e, lo, s | a ∈ Activity ∧ ad ∈ Activity-Description ∧ activity-description-ref(ad, a) ∧ e ∈ Environment ∧ lo ∈ Learning-Object ∧ s ∈ Service ∧ learning-object-ref(lo, e) ∧ service-ref(s, e) environment-ref(e, a) ∧ isvisible(ad) = "true" → isvisible(lo) = "true" ∧ isvisible(s) = "true"
10	IMS LD Specification	*Page 94 (item 6)*: "When a role is tagged to allow for the creation of new roles, the visibility rules and the users for the parent are applied to the children."
	Explanation	If the value of the attribute `create-new` of the `Role` is "allow", the values of the attribute `isvisible` of the activities related to the `Role` are the same as the values of the attribute `isvisible` of the activities related to the (sub) `Roles` of that `Role`.
	Formal Description	∀ r, r1, rp, rp1, a, a1, as | r ∈ Role ∧ r1 ⊂ r ∧ create-new(r) = "allow" ∧ rp, rp1 ∈ Role-Part ∧ role-ref(r, rp) ∧ role-ref(r1, rp1) ∧ a, a1 ∈ Activity ∧ as ∈ Activity-Structure ∧ (execution-entity-ref(a, rp) ∨ (execution-entity-ref(as, rp) ∧ execution-entity-ref(a, as))) ∧ (execution-entity-ref(a1, rp1) ∨ (execution-entity-ref(as, rp1) ∧ execution-entity-ref(a1, as)) ∧ a = a1 → isvisible(a) = isvisible(a1)
11	IMS LD Specification	*Page 16*: "When an activity-structure references one or more environments, then these will overrule the environments specified within the referenced activities."
	Explanation	The value of the attribute `isvisible` for the `Learning Objects` and `Services` of an `Environment` associated to an `Activity` must be "false" when there are `Learning Objects` and `Services` of an `Environment` associated to an `Activity Structure` in which such `Activity` is executed.
	Formal Description	∀ as, a, e, e1, e2 | as ∈ Activity-Structure ∧ a ∈ Activity ∧ execution-entity-ref (a, as) ∧ e ∈ Environment ∧ e1, e2 ∈ e ∧ environment-ref(e1, as) ∧ environment-ref(e2, a) ∧ (∃ lo1, lo2 | lo1, lo2 ∈ Learning-Object ∧ learning-object-ref(lo1, e1) ∧ learning-object-ref(lo2, e2)) → isvisible(lo2) = "false" ∀ as, a, e, e1, e2 | as ∈ Activity-Structure ∧ a ∈ Activity ∧ execution-entity-ref (a, as) ∧ e ∈ Environment ∧ e1, e2 ∈ e ∧ environment-ref(e1, as) ∧ environment-ref(e2, a) ∧ (∃ s1, s2 | s1, s2 ∈ Service ∧ service-ref(s1, e1) ∧ service-ref(s2, e2) → isvisible(s2) = "false"
12	IMS LD Specification	*Page 83 (item 12)*: "The learning-design/learning-objectives/item(s) and /prerequisites/items(s) must be accessible for all the roles, at all times in the user-interface."
	Explanation	The value of the attribute `isvisible` of the `Prerequisites` and `Learning Objectives` associated to the `Learning Design` concept must be "true".

	Formal Description	∀ ld, lg, pr | ld ∈ Learning-Design ∧ lg ∈ Learning-Objective ∧ pr ∈ Prerequisite ∧ prerequisite-ref(pr, ld) ∧ learning-objective-ref(lg, ld) ∧ isvisible(ld) = "true" → isvisible(lg) = "true" ∧ isvisible(pr) = "true"

The role axioms (Table 4) constrain how the instances of the Role concept must be created in the learning design. Thus, depending on the values of the Role attributes (match-persons, min-persons, and create-new), a number of instances of that concept could be created (axioms 14 and 15). Furthermore, there are instances that restrict how the Role instances can be used in the definition of the other learning elements (axiom 16). The role axioms are also applied to the subclasses of Roles that are typically created to specify different categories of teachers or users of the Services managed the Environments associated to the Execution Entities of the learning design.

Table 4. Formal definition of the role axioms

13	IMS LD Specification	*Page 25 (item 0.2.6)*: "Specifies the minimum number of persons bound to the role before starting a run. When the attribute min-persons and max-persons are empty, there are no restrictions. When used, the following rule applies: 0 <= min-persons <= max-persons."
	Explanation	The value of the attribute `max-persons` of a `Role` must be greater than the value of the attribute `min-persons` of that `Role`.
	Formal Description	∀ r | r ∈ Role → max-persons(r) ≥ min-persons(r)
14	IMS LD Specification	*Page 25 (item 0.2.1)*: "This attribute [create-new] indicates whether multiple occurrences of this role may be created during runtime. When the attribute has the value "not-allowed" then there is always one and only one instance of the role."
	Explanation	If the value of the attribute `create-new` of a `Role` is "not allowed", there is a unique instance of that `Role`.
	Formal Description	∀ r | r ∈ Role ∧ create-new(r) = "not-allowed" → ¬ ∃ r1 | r1 ∈ r
15	IMS LD Specification	*Page 25 (item 0.2.4)*: "This attribute is used when there are several sub roles (e.g. chair, secretary, member). Persons can be matched exclusively to the sub roles, meaning that a person, who has the role of chair, may not be bound to one of the other roles at the same time."
	Explanation	If a `Role` has (sub) `Roles` and the value of the attribute `match persons` of that `Role` is "exclusively-in-roles", the (sub) `Roles` must be disjoint.
	Formal Description	∀ r, r1, r2, p1, p2 | r ∈ Role ∧ match-persons(r) = "exclusively-in-roles" ∧ r1, r2 ∈ r ∧ p1, p2 ∈ Person ∧ p1 ∈ r1 ∧ p1 ∈ r1 → p1 ≠ p2
16	IMS LD Specification	*Page 90*: "The same role can be associated with different activities or environments in different role-parts, and the same activity or environment can be associated with different roles in different role-parts. However, the same role may only be referenced once in the same act. If multiple activities or environments need to be associated for the same role an activity-structure or wrapper environment should be used."
	Explanation	For a same `Act`, a given instance of a `Role` can just appear once in the `Role Parts` executed in the context of that `Act`.
	Formal Description	∀ a, r, rp | a ∈ Act ∧ r ∈ Role ∧ rp ∈ Role-Part ∧ role-part-ref(rp, a) ∧ role- ref(r, rp) → ¬ ∃ rp1 | rp1 ∈ Role-Part ∧ rp1 ≠ rp ∧ role-part-ref(rp1, a) ∧ role-ref(r, rp1)

Finally, the activity axioms (Table 5) constrain the relations between the Execution Entities and the other components of the learning design. Thus, there are axioms to determine what Roles are involved in the Support Activities (axioms 17 and 18), to restrict the values of the attributes of the Activity Structure concept (axiom 19), and to constrain the relation between the Acts and the Execution Entities of a given design (axiom 20).

Table 5. Formal definition of the activity axioms

17	IMS LD Specification	*Page 29 (item 0)*: "When the optional role-ref element is set, it is expected that the support activity will act for every single user in the specified role(s). That is: the same support activity is repeated for every user in the role(s)."
	Explanation	If a `Support Activity` has assigned a `Role` (that is, the attribute `role-ref` has a value), this activity will be executed by all the instances of the subclasses of such `Role`. On the other hand, it is necessary to define a `Role-Part` for each subclass of the `Role` that is applied to the `Support Activity`.

	Formal Description	∀ a, rp, sa, r, p | a ∈ Act ∧ rp ∈ Role-Part ∧ role-part-ref(rp, a) ∧ sa ∈ Support-Activity ∧ r ∈ Role ∧ role-ref(r, sa) → ∃ r1, rp1 | rp1 ∈ Role-Part ∧ role-part-ref(rp1, a) ∧ r1 ∈ r ∧ role-ref(r1, rp1) ∧ execution-entity-ref(sa, rp1) ∀ sa, r, r1 | sa ∈ Support-Activity ∧ r ∈ Role ∧ role-ref(r, sa) ∧ r1 ∈ r → role-ref(r, rp)
18	IMS LD Specification	Page 29 (item 0): "When the role-ref is not available, the support activity is a single activity (like the learning-activity)."
	Explanation	If a `Support Activity` has not assigned a `Role` (that is, the attribute `role-ref` has not a value), it will be considered as a simple `Activity`, and there would not be applied to every instance of the `Role`.
	Formal Description	∀ rp, sa, r, as | rp ∈ Role-Part ∧ sa ∈ Support-Activity ∧ as ∈ Activity-Structure ∧ r ∈ Role → ¬ ∃ r1 | r1 ∈ r ∧ role-ref(r1, rp) ∧ (execution-entity-ref(sa, rp) ∨ (execution-entity-ref(as, rp) ∧ execution-entity(sa, as)))
19	IMS LD Specification	*Page 31 (item 0)*: "When the attribute 'number-to-select' is set, the activity-structure is completed when the number of activities completed equals the number set. The number-to-select must be the same as or smaller than the number of activities (including unit-of-learnings) which are at the immediate child level."
	Explanation	The value of the attribute `number to select` of the `Activity Structure` must be smaller than the number of the `Activities` of that `Activity Structure`. To express this axioms it is necessary to define a variable that is the number of `Activities` of which an `Activity Structure` is composed of.
	Formal Description	∃ na ∈ Integer | na = 0 → ¬ ∃ as, a | as ∈ Activity-Structure ∧ a ∈ Activity ∧ execution-entity-ref(a, as) ∀ as, a | as ∈ Activity-Structure ∧ a ∈ Activity ∧ execution-entity-ref(a, as) → na = na + 1 ∀ as | as ∈ Activity-Structure ∧ number-to-select(as) ≤ na
20	IMS LD Specification	Page 76: "Each role-part associates exactly one role with exactly one type of activity (including the performance of another unit-of-learning and activity-structures), or with one environment (equivalent to an organization in Content Packaging)."
	Explanation	In an `Act` must exist at least an `Role Part` that has assigned an `Activity` or `Activity Structure`. If this axiom is not included in the ontology specification, it could be possible that all the `Role Parts` of an `Act` establish relations with the `Environment`, which is not an `Execution Entity` and, therefore, does not define any ending condition (through the `when-role-part-completed` or `time-limit` attributes).
	Formal Description	∀ act | act ∈ Act → ∃ rp, as, a | rp ∈ Role-Part ∧ role-part-ref(rp, act) ∧ as ∈ Activity-Structure ∧ a ∈ Activity ∧ (execution-entity(a, rp) ∨ (execution-entity(as, rp) ∧ execution-entity(a, as)))

Modelling Issues of the Learning Design Ontology

The learning design ontology has been developed for semantically describing every element of the IMS LD specification, solving the drawbacks of its representation in XML-Schema. In this development, new classes (or concepts) were introduced with the aim of improving the modelling of the IMS LD elements, but these classes do not add new learning elements that would extend the IMS LD specification (such as an ontology of educational organizations). In fact, it would be possible to carry out a straightforward translation from the XML-Schema representation of the IMS LD into the learning design ontology, and vice versa. Considering this, these new classes would be translated in the following way: The Execution-Entity, Complete-Unit and Item classes are abstract, which means that they are not instantiated when a learning design is specified. Therefore, these three classes will not be translated, because they are not considered as part of a given learning design, and they were introduced in the LD ontology to improve the structure of the taxonomy by taking advantage of the inheritance and subsumption mechanisms (particularly useful for description logic reasoners).

All the subclasses of the abstract classes are directly related to an XML-Schema element, or groups of elements, that represents an entity of the IMS LD specification: both ontology classes and XML-Schema elements have the same attributes, which means that the translation between them will be straightforward. Nevertheless, in the translation from the ontology classes into XML-Schema, there exist semantic (or knowledge) loss because the ontology classes are more expressive.

To improve the semantic description of the taxonomy concepts of the LD ontology and increase the reasoning capabilities of the ontology, taxonomic (such as disjoint and exhaustive) and mathematical (such as inverse, symmetric and transitive) properties of relations have also been added to the LD ontology. However, these properties cannot be considered extensions of the IMS LD specification, because they do not introduce new elements to the learning design.

Finally, the axioms formulated in the LD ontology are the formal specification of the constraints among the elements of a learning design. In the IMS LD specification these constraints are expressed in natural language,

while in the LD ontology they are represented in first order logic. Therefore, the axioms of the ontology are not an extension of the IMS LD specification either: they are the *same* entities represented in a different way.

Modelling Example: Description of the Jigsaw Methodology

Cooperative learning is a technique in which learning is achieved by means of group activities. In this type of learning, the acquisition of knowledge and skills comes about as the result of the interaction among groups, being based on aspects such as individual responsibility, positive interdependence (each individual depends on all the others in order to succeed) and the development of the interpersonal abilities that are necessary in real life situations. With the use of cooperative learning techniques, educational processes may obtain many benefits from the perspective of motivation, cognoscence and social cohesion (Sub et al., 1999). Broadly speaking, cooperation-based teaching/learning processes can be organized according to the following procedures:

1. The didactic objectives are presented to the students.
2. An initial assessment is made.
3. The objectives of each group are defined.
4. The content and evaluation criteria are explained in detail.
5. The groups carry out the activity.
6. The students evaluate each other's work.
7. The professor provides an individual evaluation.
8. Each group is evaluated.

Those groups that failed on achieving the objectives are reorganized.

There are various didactic techniques aimed at establishing co-operational relationships during learning activities: jigsaws, investigation groups, STAD (Student Team-Achievement Divisions) technique, TGT (Teams-Games Tournaments) and peer tutorship. To demonstrate the application of the IMS-LD ontology, the Jigsaw methodology has been chosen (Figure 5). Besides its popularity, this technique is suitable to understand how cooperation-based learning activities are carried out in activities that can easily be broken down into their constituent parts. In short, the Jigsaw follows the general procedure described above, organizing the students into groups of 4 or 5 individuals, and assigning different learning material to each member of the group, so that each student receives a fragment of the information of any given topic that is the matter of the study. Each member prepares the topic with his personal material, joining up with the members of the other groups who have the same topic. After that, he returns to his own group in order to explain and discuss the topic along with the other members.

As it can be seen in Figure 5, the Method representing the Jigsaw approach has a Play made up of a set of Acts, which are executed in sequence. Each Act comprises a set of concurrent Role Parts that relate roles (professor, student, etc.) with activities. The Complete Act concept is used to control the end of an Act, either stating the maximum time for the realization of the Act, or the associated Role Parts that must be completed. For example, the First-Moment Act refers to the activities to be performed during the initial stage of the Jigsaw (corresponding to the general steps 1-4 described above) and is made up of two Role Parts (Ap_Work and In_Att). A relation jsfm-wrpc:when-role-part-completed between the instances jsfm-cp of Complete Act and In_Att of Role Part is used to indicate when this Act is completed. In this Act, the professor carries out the following activities with the groups: objectives presentation, in which he explains the topic of the subject; student assessment, in which an assessment in order to verify the students' prospects and needs is carried out; group creation, in which the activities to be realized and the rules and criteria of the evaluation are explained. These activities are represented by a series of Support or Learning Activities that are grouped into Activity Structures. Each Role Part associates a single Support Activity, Learning Activity or Activity Structure to a certain Role, as it is shown in Figure 6. An activity Structure is completed when the value of the number-to-select attribute is equal to the number of the activities in the set that have been completed. In the example shown in Figure 6, Ap-Wrk-AS will be completed when 6 activities have been completed.

The two `Acts` following the *First-Moment* are the key moments in the Jigsaw technique. The *Jigsaw Part 1* comes when the groups are formed with students with the same study topic. Each topic is a part of the whole subject to be studied, around which the students carry out the activities according to the information that has been supplied by the professor. In the *Jigsaw Part 2* each group comprises students that, collectively, possess all the study material. In this `Act`, each student explains the topic that he has previously studied to the rest of the group. In parallel, the professor monitors and guides the activities carried out by each group.

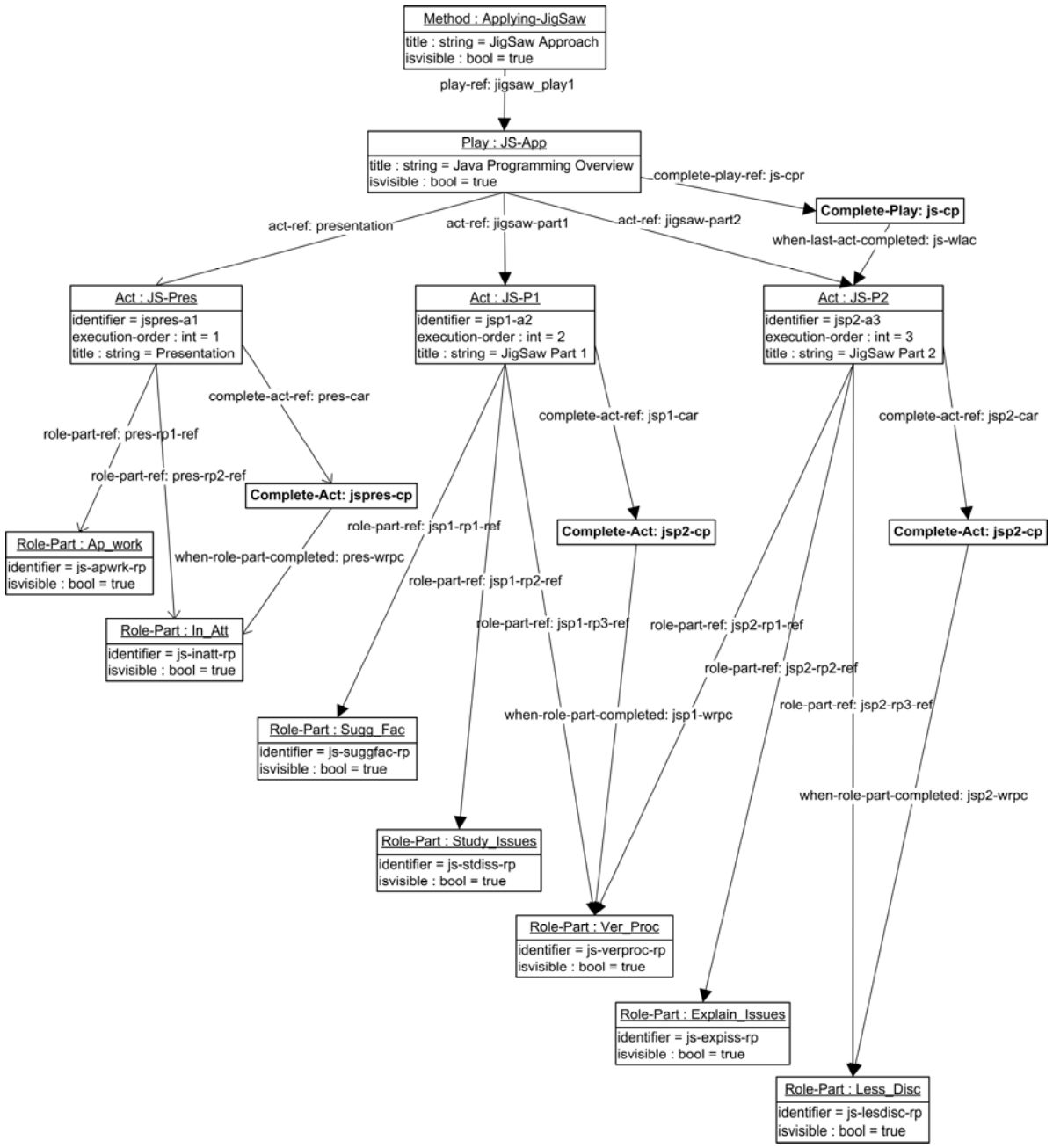


Figure 5. Definition of the Method, Plays and Acts of the Jigsaw example

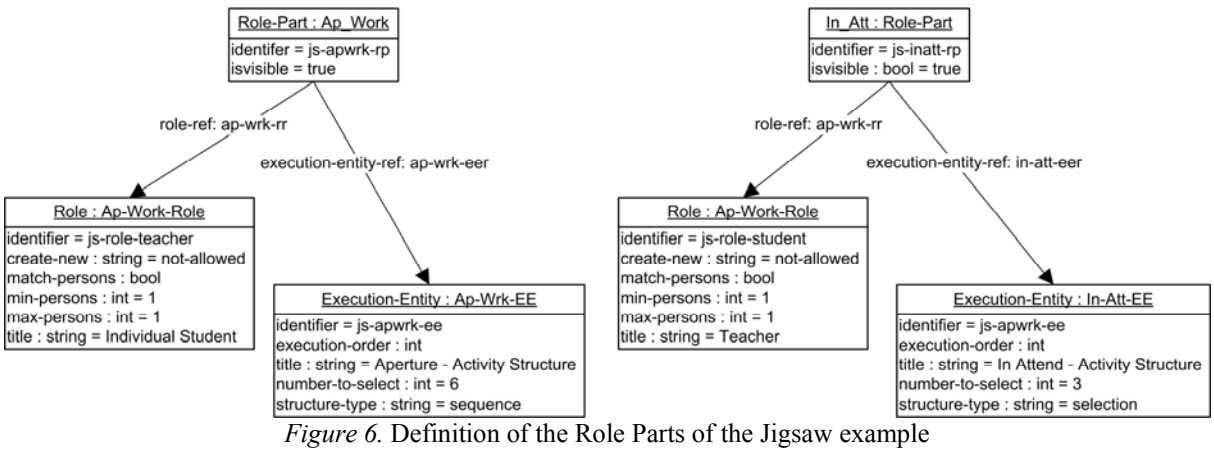


Figure 6. Definition of the Role Parts of the Jigsaw example

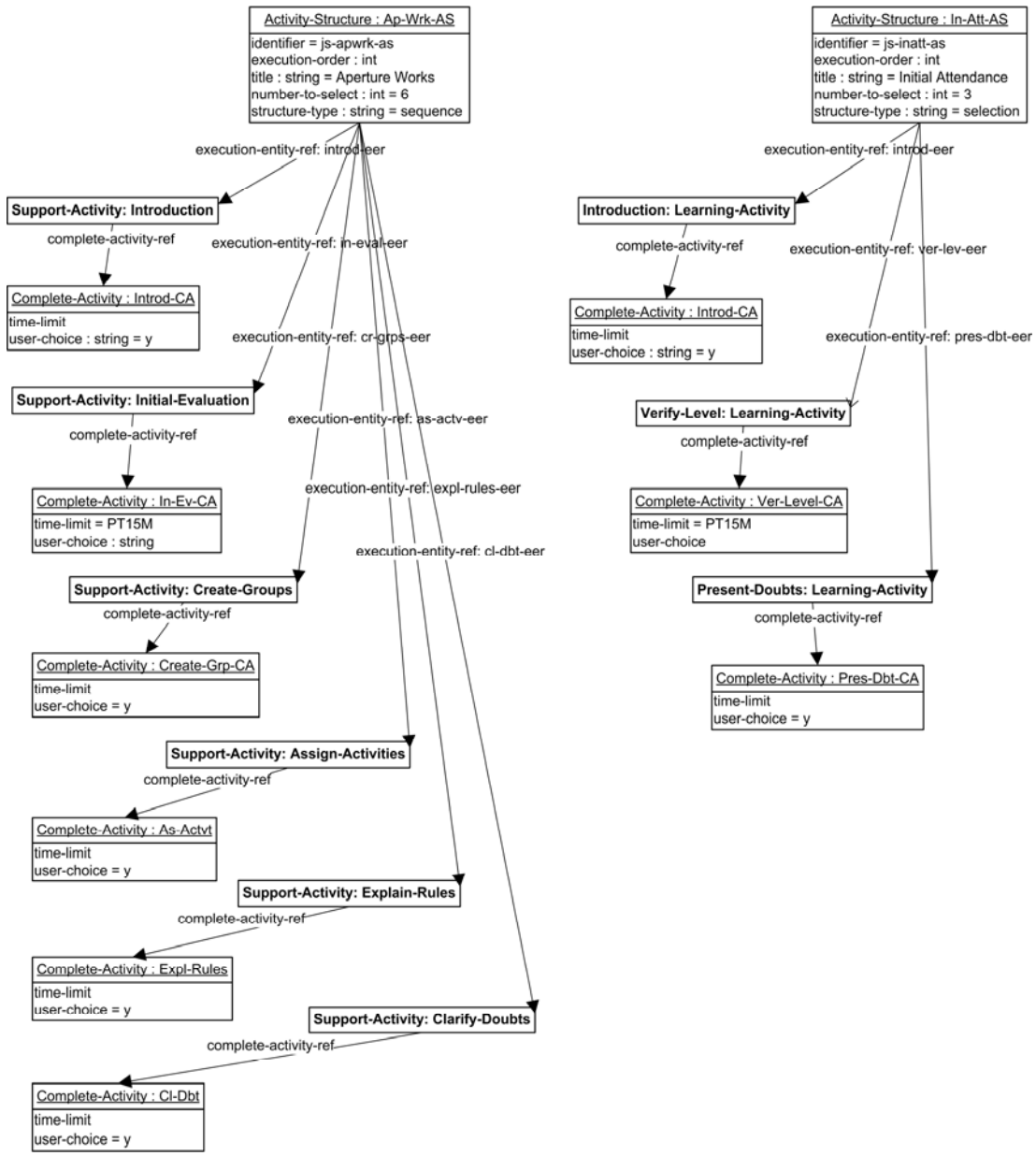


Figure 7. Definition of the Activity Structures of the Jigsaw example

The Acts and Role-Parts shown in Figures 5 and 6 partially represent the dynamics of the educational process. Using Activity-Structures this dynamics can be described in much more detail, according to workflows determining the way in which Support and Learning Activity will be executed in an educational process. Figure 7 shows the order and conditions in which the activities of the professor and the groups are carried out in the First-Moment Act.

The order in which the activities in an Activity Structure are carried out is determined by means of the execution-order and structure-type attributes. The Activity Structure Ap-Wrk-AS, associated with the professor, presents a structure-type attribute with value "sequence", indicating that the professor executes the Support Activity in a sequence, while the Activity Structure In-Att-AS, related to the groups, shows a "selection" value, indicating that the order of execution depends on a parameter. In this case, the order of the students' activities depends on the professor. Using the Complete Activity concept, activities may be completed by a decision of the role, or at the end of a given time. The support activity Introduction is completed with a decision made by the professor (setting the user-choice attribute), and the Support Activity Initial-Evaluation is completed after the completion of a given time interval (setting the time-limit attribute). The Support Activities Introduction and Explain-Rules determine the beginning of the Learning Activities Introduction and Clarify-Doubts. When an activity is finished, an action to be carried out can be indicated with the Completion Unit and Feedback Description concepts, which are related by the completion-ref relation. For example, with these concepts, a

resource can be assigned to present information regarding activity feedback. This may be highly useful when considering the individual reforming of those groups that did not attain the objectives, or when considering future activities.

Associated to the Learning Activities, the Learning Objectives aid the professor, along with the students, to evaluate the learning process. For example, the objective for the Learning Activity, Verify-Level in the First-Moment Act, would be to verify the students' level of knowledge in order to be able to subsequently define the group in a homogeneous manner. In this case, this Learning Activity may consist of a 50-question, multiple-choice questionnaire for which the students have 25 minutes. As a way of homogenizing the groups, the professor would be able to exempt those students with marks of 85% or over from the Jigsaw activities (Figure 8).

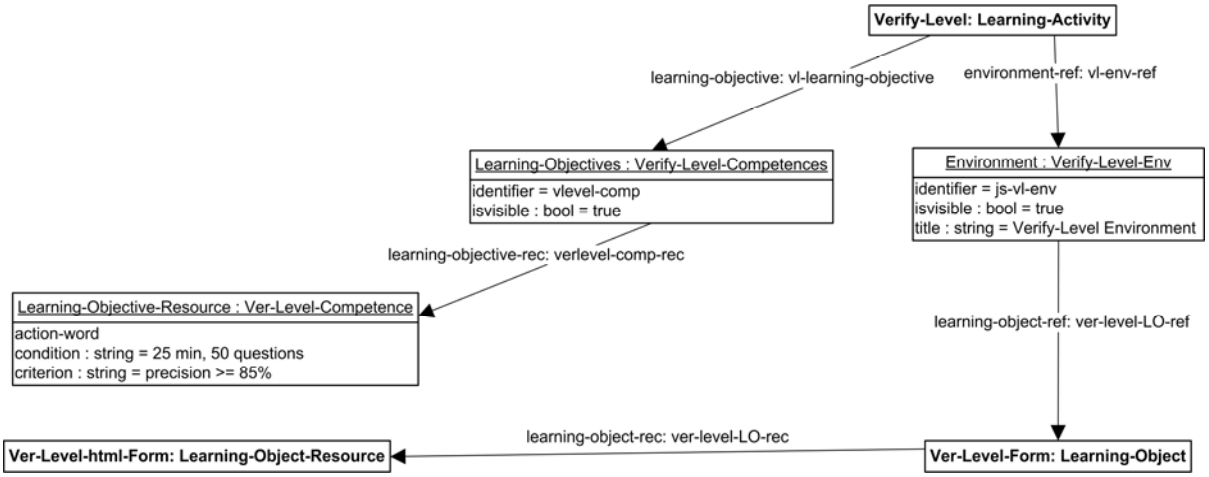


Figure 8. Definition of a learning activity of the Jigsaw example.

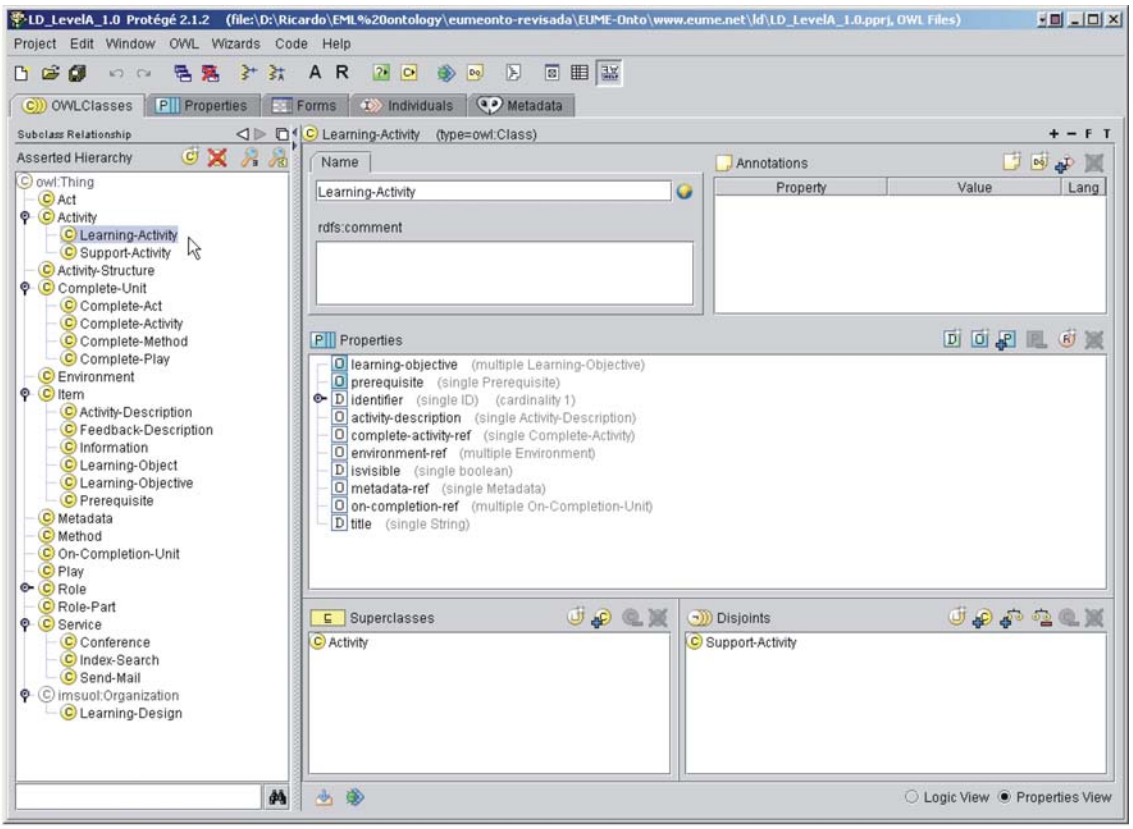
Construction and Use of the IMS LD Ontology

For the construction of the ontology we have used Protégé, an extensible, platform-independent environment for creating and editing ontologies and knowledge bases (Noy et al., 2000). As is shown in Figure 9, after describing the ontology at the knowledge level, the Protégé-OWL plug-in was used to export it to the OWL language (Dean & Schreiber, 2004) language, which is the W3C recommendation for the Semantic Web. The OWL specification of the ontology can be downloaded from *http://www.eume.net/ontology/imsld_a.owl*.

The IMS LD ontology is currently used in the EUME system, a learning management system oriented to support educational activities in the classroom (Sanchez et al., 2003), as a common language to manage the information about the educational resources available in the environment. The software architecture of the system is based on intelligent agent technology, and follows a multi-layer topology (Riera et al., 2004) with four different tiers: the Resource tier is responsible for low-level tasks (control of hardware/software); the Services tier is responsible for the educational activities; the Mediator tier is a common channel that routes every message between services and clients; and, finally, the Client tier contains graphic interfaces that allow the adaptation/personalization of contents/services to the learning environment and user preferences. The multi-agent EUME System was implemented in JADE, a FIPA-compatible middleware that facilitates both agent implementation and agent communication through message passing mechanisms.

The EUME agents use the IMS LD ontology as a common language to manage the information about the educational resources available in the environment. This is done by means of a set of JADE classes that were implemented to enable the agents *(1)* to manage the OWL code, and *(2)* then to generate messages in accordance with the FIPA-SL style sheet. Figure 10 illustrates the mechanism by describing how the description of a certain *Activity* is requested. The client agent (A/C) defines a template (FIPA-SL) according to this request, which is sent to the service agent (Search A/S) in the Services tier. After that, this agent generates another template to communicate with the specific resource to access the database.

1



2

```
<owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://www.eume.net/ontology/uol.owl"/>
   <owl:imports rdf:resource="http://www.eume.net/ontology/lom.owl"/>
</owl:Ontology>
<owl:Class rdf:ID="Method">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
       <owl:ObjectProperty rdf:ID="play-ref"/>
     </owl:onProperty>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">The method
   </rdfs:comment>
  </owl:Class>
</owl:Class>
<owl:Class rdf:ID="Play">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:minCardinality>
     <owl:onProperty>
       <owl:ObjectProperty rdf:ID="act-ref"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
```

Figure 9. Protégé and OWL description of the IMS LD ontology

The EUME system is intended to facilitate the design and realization of different learning activities. The process begins with the professor specifying the learning design by using the out-of-classroom interface, a web interface that enables the introduction of Units of Learning and their corresponding Methods, Plays as well as other learning design elements. Figure 11 shows a screenshot of this web interface to illustrate the introduction of a number of Activities (Introduction, Verify-level and Present-Doubts) associated to the Presentation Act of a Java Programming Overview Play. For each Activity, a Powerpoint file was selected as a Learning Object resource. Once this design stage is completed, the learning activities are ready to be used in the classroom. Here, the

professor uses a PDA interface, which contains the agent client to control the available hardware and software resources as well as to access the learning elements previously designed. After logging into the system (Figure 12 A), the EUME agents automatically retrieve the information from the database and show the Activities associated to the current Java Programming Overview Play (Figures 12 B-C).

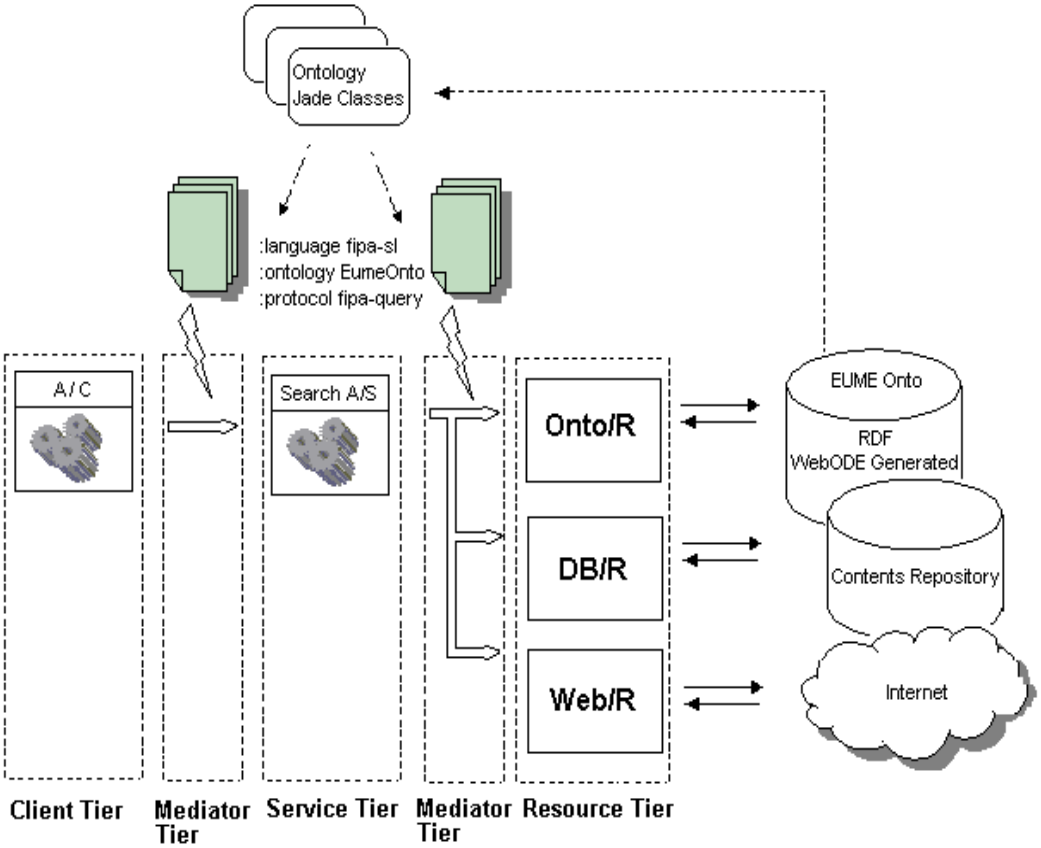


Figure 10. Mechanism to request an Activity in EUME

Finally, the LD ontology could be also used in a learning management system that implements the IMS LD specification following the XML-Schema language. In such a case, a translator from the ontology (expressed in OWL) into the XML-Schema representation, and vice versa, would be required. For instance, in a service oriented architecture, the translation operation could be offered by a Web service that would receive SOAP messages whose content is (part of) the ontology to be translated, and would send the result of the translation.

Discussion and Future Work

The IMS LD specification is expressive enough from the point of view of the learning process designers. Nevertheless, the informal specification of the IMS information and behavioural models increases the complexity of the IMS LD to be understood by programmers, as they are not usually educational specialists. This complexity could provoke misinterpretations and, even, errors when the IMS LD specification is incorporated to the development of applications.

The XML-Schema language is not enough expressive to represent al the knowledge compiled in the three models of the IMS LD specification. Mainly, hierarchical taxonomies, relation properties, and semantic constraints between the learning design elements cannot be represented in XML-Schema. To solve these limitations, the software system used to design and execute the unit of learning could codify the semantics of the specification in the programming language in which it is developed. This strategy has been followed by the Reload (JIST, 2004) and CopperCore (Vogten & Martens, 2005) environments to allow to users the design and execution of a unit of learning respectively. However, the main drawback of this approach is that the software programs are not *easy to maintain*, because of if the IMS LD specification is modified, it would be needed to re-codify the programs for including such modifications.

These two issues are solved with the learning design ontology. On one hand, as the semantics of the concepts is *precisely* defined, there should be no misinterpretations or errors when the instances of the concepts are created and managed in runtime. In this sense, new concepts, attributes/relations, and formal axioms have been identified and formalized in the ontology. It is important to emphasize that these add-ons neither change nor extend the IMS LD specification, but they *enrich* the description of the semantics of the IMS LD elements. On the other hand, as the semantics of the IMS LD specification is *explicitly* described, it is not necessary to codify such semantics in the development of the software program that allow to users to design and execute the unit of learning. Thus, general reasoner following the logic paradigm associated to the language in which the ontology is represented can be used to check the consistence of the unit of learning in both the design and runtime phases. For example, we can use a reasoner in description logic (like Pellet or Racer) to carry out inferences with the learning design ontology expressed in OWL.

The main drawback of the LD ontology is focused on the limitations of the expressiveness and reasoning capabilities of the ontology representation languages. For example, if the goal of an application is to guarantee that the edition and execution of the learning design satisfies the IMS LD specification, the OWL language would not be the best choice as it currently does not support the definition of axioms that check constraints between concepts. However, OWL could be an appropriate language for solving the interoperability issues between applications.

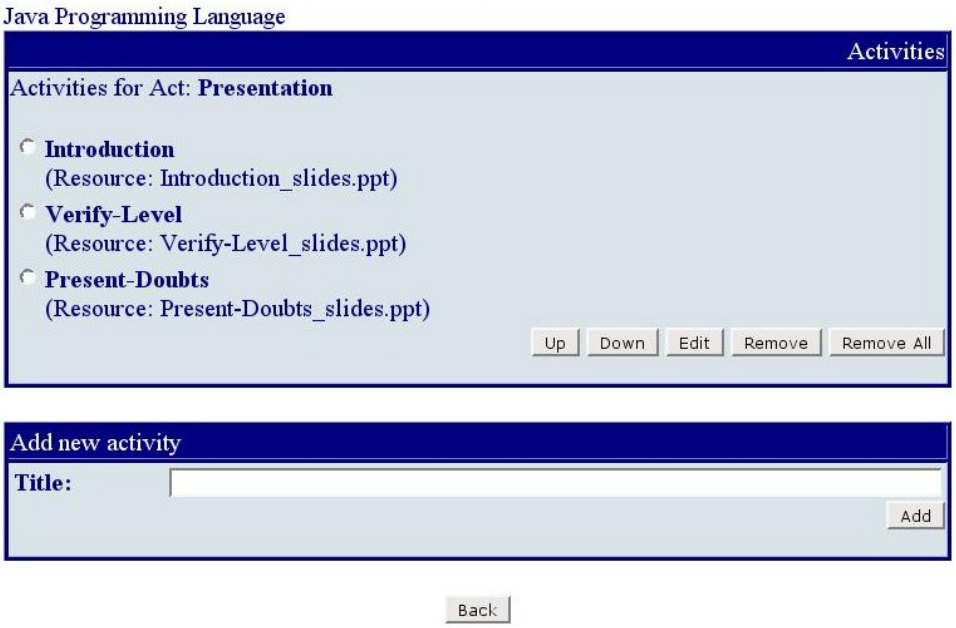


Figure 11. Interface used to define the Activities of the Jigsaw example.

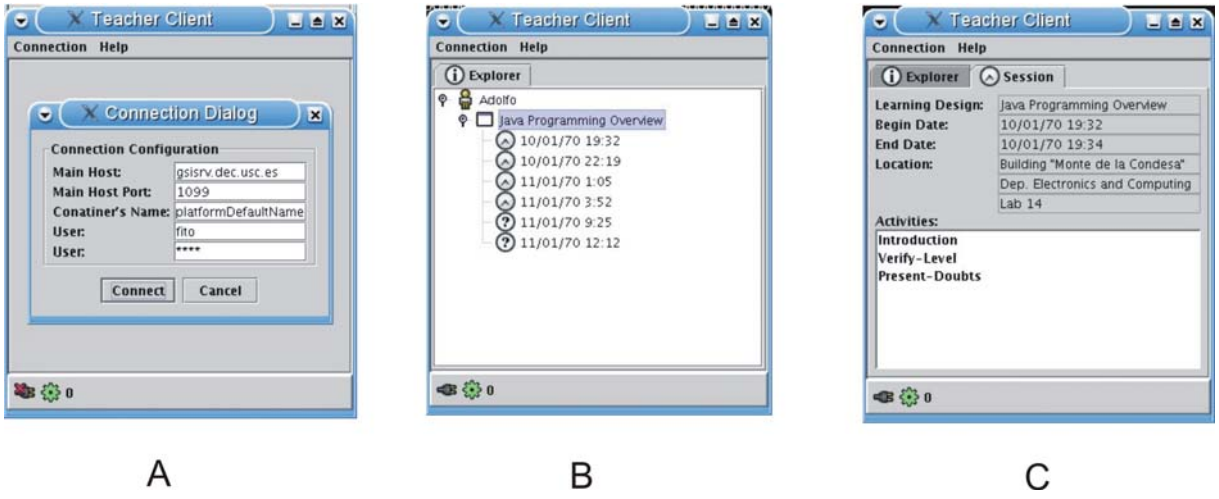


Figure 12. In classroom PDA interfaces.

As future work we have planned to translate the ontology axioms into SWRL (Horrocks et al., 2004), which is the language currently proposed to express restrictions in OWL. On the other hand, we are working on the extension of the ontology to include the concepts and axioms of the levels B and C of the IMS LD specification.

Acknowledgments

The authors would like to thank the Xunta de Galicia for their financial support in carrying out this work under the project PGIDT02TIC20601PR.

References

Barros, B., Verdejo, F., Read, T., & Mizoguchi, R. (2002). Applications of a Collaborative Learning Ontology. *Paper presented at the Second Mexican International Conference on Artificial Intelligence (MICAI 2002)*, April 22-26, 2002, Yucatan, Mexico.

Brase, J., & Nejdl, W. (2004). Ontologies and Metadata for eLearning. In Staab, S. & Studer, R. (Eds.), *Handbook on Ontologies*, Berlin: Springer-Verlag, 555-574.

Brickley, D. (1996). *Towards an open question-interchange framework*, retrieved October 31, 2005 from http://www.ilrt.bris.ac.uk/netquest/about/lang/motivation.html.

Dean, M., & Schreiber, G. (2004). *OWL – Web Ontology Language Reference*, retrieved October 31, 2005 from http://www.w3.org/TR/owl-ref.

Dublin Core Metadata Initiative (2003). *Dublin Core Metadata Element Set, Version 1.1. Reference Description*, retrieved October 31, 2005 from http://dublincore.org/documents/dces.

Gil, Y., & Ratnakar, V. (2002). A Comparison of (Semantic) Markup Languages. *Paper presented at the 15th International FLAIRS Conference*, May 16-18, 2002, Pensacola Beach, Florida, USA.

Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2004). *Ontological Engineering*, Berlin: Springer.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, 5*, 199-220.

Horrocks, I., Pater-Schneider, P., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, retrieved October 31, 2005 from. http://www.w3.org/Submission/SWRL.

IMS Global Learning Consortium (2003a). *IMS Learning Design Information Model. Version 1.0 Final Specification*, retrieved October 31, 2005 from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html.

IMS Global Learning Consortium (2003b). *IMS Learning Design Binding. Version 1.0 Final Specification*, retrieved October 31, 2005 from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bindv1p0.html.

Inaba, A., Tamura, T., Ohkubo, R., Ikeda, M., Mizoguchi, R., & Toyoda, J. (2001). Design and Analysis of Learners Interaction based on Collaborative Learning Ontology. *Paper presented `at the 2nd European Conference on Computer-Supported Collaborative Learning (Euro-CSCL'2001)*, March 22-24, 2001, Maastricht, The Netherlands.

JIST - Joint Information Services Committee (2004). *RELOAD Editor - Introductory manual*, retrieved October 31, 2005 from http://www.reload.ac.uk/ex/editor_v1_3_manual.pdf.

Kabel, S., Wielinga., B., & de How, R. (1999). Ontologies for indexing Technical Manuals for Instruction. *Proceedings of the AIED-Workshop on Ontologies for Intelligent Educational Systems*, LeMans, France, 44-53, retrieved October 15, 2005, from http://www.ei.sanken.osaka-u.ac.jp/aied99/a-papers/SC-Kabel.pdf.

Kiefer, M., Lausen, G., & Wu, J. (1995). Logical Foundations of Object-Oriented and Frame-Based Languages *Journal of ACM, 42*, 741-843.

Koch, M. (2002). Interoperable Community Platforms and Identity Management in the University Domain. *International Journal on Media Management, 4* (1), 21-30.

Koper, R. (2001). *Modelling units of study from a pedagogical perspective the pedagogical meta-model behind EML*, retrieved October 31, 2005 from http://dspace.learningnetworks.org/retrieve/33/ped-metamodel.pdf.

Noy, N., Fergerson, R., & Musen, M. (2000). The knowledge model of Protege-2000: Combining interoperability and flexibility. *Paper presented at the 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW),* October 2-6, 2000, Juan Les Pins, France.

Rawlings, A., van Rosmalen, P., Koper, R., Rodríguez-Artacho, M., & Lefrere, P. (2002). Survey of Educational Modelling Languages. *CEN/ISSS WS/LT Learning Technologies Workshop*, retrieved October 15, 2005 from http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/activity/emlsurveyv1.pdf.

Riera, A., Sánchez, E., Lama, M., Amorim, R., Vila, X., & Barro, S. (2004). Study of Communication in a Multi-Agent System for Collaborative Learning Scenarios. *Paper presented at the 12th Euromicro Conference on Parallel, Distributed and Network based Processing (PDP2004)*, February 11-13, 2004, Coruña, Spain.

Rodríguez-Artacho, M., Verdejo, F., Mayorga, J., & Calero, M. (1999). Using a High-Level Language to Describe and Create Web-Based Learning Scenarios. *Paper presented at the IEEE Frontiers in Education Conference (FIE)*, November 10-13, 1999, San Juan, Puerto Rico USA.

Sánchez, E., Lama, M., Amorim, R., Riera, A., Vila, X., & Barro, S. (2003). The EUME Project: Modelling and Design of an Intelligent Learning Management System. In *Proceedings of the AIED-Workshop on Intelligent Learning Management Systems*, Sydney, Australia, 183-191, retrieved October 15, 2005 from http://www-gsi.dec.usc.es/~eume/publications/aied03.pdf.

Slavin, R. (1995). *Cooperative learning: Theory, research, and practice*, Boston: Allyn & Bacon.

Sub, C., Burkhard, F., & Brössler, P. (1999). Metamodeling for Web-Based Teachware Management. *Paper presented at the International WWWCM-Workshop on the World-Wide Web and Conceptual Modeling*, November 15-18, 1999, Paris, France.

Thompson, H., Beech, D., Maloney, M., & Mendelsohn, N. (2004). *XML-Schema Part 1: Structures Second Edition*, retrieved October 31, 2005 from http://www.w3.org/TR/xmlschema-1.

Verbert, K., & Duval, E. (2004). Towards a global architecture for learning objects: a comparative analysis of learning object content models. *Paper presented at the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA*), June 21-26, 2004, Lugano, Switzerland.

Vogten, H., & Martens, H. (2005). *CopperCore 2.2.2*, retrieved October 31, 2005 from http://www.coppercore.org.