Graphical Ontology Modeling Language for Learning Environments

GILBERT PAQUETTE*

LICEF/CIRTA Research Center, Télé-université, Montreal

In the last fifteen years, our main goal has been to synthesize and combine various forms of graphical representations that are useful for educational modeling and knowledge management, using an integrated graphical formalism. We have shown that very different kinds of representation, conceptual maps, flowcharts, decision trees and others, can all be modeled more precisely, using the MOT graphic language based on typed objects (concept, procedures, principles, facts) as well as few typed links. With this set of primitive graphic symbols, it has been possible to build very different graphic models, from simple taxonomies to ontologies, more or less complex learning designs, delivery process, decision systems, and methods.

Recent developments have led to two specialisations of the graphic language. The first one is a powerful, yet simple graphic language to build ontologies for a knowledge domain. The second one enables to model learning designs and scenarios in a standardized and computable way. The association between both kinds of models specifies the central part of a learning environment at the design phase, and enables its delivery to learners and educators. In the final section, I assert that knowledge representation for education should be graphic, user-friendly, general, scalable, declarative, standardized and computable. A discussion of these criteria concludes.

Keywords: Knowledge representation, knowledge modeling, ontologies, description logic, learning design, graphic languages.

*Corresponding author: E-mail: gilbert.paquette@teluq.uqam.ca
INTRODUCTION

Building high quality learning designs is a very important and demanding task that we started to address in the beginning of the nineties by progressively building an instructional engineering method, a delivery system and a graphical knowledge modeling editor.

I take here the point of view that the design of Learning Environments is the result of a knowledge engineering process, where knowledge and competencies, learning designs and delivery processes are represented using a graphic formalism, executable by a computer or not, and constructed based on an integrated framework.

The graphic representation formalism that I present here (Paquette 1996, 2002, 2006) has evolved and has been tested for many years in a vast array of modeling applications in various contexts. It is used by trainers for corporate training. Designers or professors use it to prepare university courses or to propose modeling exercises to their students. It has served to model processes for the introduction of IT in a computer-supported high school, or to model methods or research projects processes.

In the first introductory section I discuss the basis for our educational representation system, from a software engineering, an instructional design and a cognitive science perspective.

In the second section I summarize our initial efforts to create a generic graphical modeling language. The theoretical and experimental foundation of MOT (Modeling using Object Types) will be presented. The language and its editor presented here have served as the backbone of our MISA instructional design methodology (Paquette et al. 1994, 2003, 2005), to model informally the main components of a learning system. It has also served to develop the method itself as a precise multi-actor process to support an instructional design team. Even at this informal level, the language constitutes an important tool for precise definition and communication.

In the third and central section I present a specialization of the graphic language to MOT+OWL, to represent domain knowledge and competencies as ontologies, thus bringing the language at a formal and computational level.

In the fourth section, I will present the MOT+LD editor, a graphical tool to construct IMS Learning Designs, and present the concept of semantic annotation of learning design components. Together, a MOT+LD model, a MOT+OWL domain model and the association between their components provide adequate specification power to build functional learning environments in any domain of study.
In the last section, I will discuss the pros and cons of this proposal according to a set of criteria.

BASIS FOR AN EDUCATIONAL REPRESENTATION LANGUAGE

When designers start building a learning environment, two basic questions arise: “Which knowledge must be acquired?” and “How should the activities and the environment be organized to best achieve knowledge and competency acquisition? To help designers answer these questions, we have developed a general graphical modeling method and some editing tools. In this section, we briefly present the experimental and theoretical basis for this modeling language.

Knowledge Types Integration

It is often said that a picture is worth a thousand words. That is true of sketches, diagrams, and graphs used in various fields of knowledge. Conceptual maps are widely used in education to represent and clarify complex relationships between concepts, and to facilitate knowledge construction by the learners or environment design by educators. Flowcharts are graphical representations of procedural knowledge or algorithms, composed of actions and decisions that trigger series of actions in a dynamic rather than static way. Decision trees constitute another form of representation used in various fields, particularly in decision-making expert systems, establishing influence or cause/effect relationships between various factors; decision trees have been proven equivalent to building a set of rules that constitute the knowledge base of the expert system.

All these representation methods are useful at an informal level, as thinking aids and tools for the communication of idea, but they have limitations. One is the imprecise meaning of the links between the entities that compose the model. Another one is the ambiguity in graphs where objects, actions on objects and statement of properties about them are all mixed-up and are not represented as such. It is important to be able to declare transparently the types of the entities involved in a graph, especially if we have educational applications in mind.

Another difficulty is the impossibility to combine more than one representation in the same model. For example, concepts used in procedural flowcharts as entry, intermediate or terminal objects could be given a more precise meaning by developing them in conceptual sub-models of the procedure. The same is true of procedures present in conceptual models that could be developed as procedural sub-models described by flowcharts, combined or not with decision trees.
In software engineering, many graphic representation formalisms have been or are used such as Entity-Relationship models (Chen, 1976), Modern structured analysis (Yourdon, 1989), Conceptual Graphs (Sowa, 1984), Object modelling technique (OMT) (Rumbaugh et al, 1991), KADS (Scheiber et al, 1993), or Unified Modeling Language UML (Booch et al, 1999). These representation systems have been built for the analysis and architectural design of complex information systems. The most recent ones require the use of up to eight different kinds of model and the links between them are hard to follow without considerable expertise.

Our goal is different. We need a graphic representation system that is both simple enough to be used by educational specialists who are not in general computer scientists, let general and powerful enough to represent the components of educational environments and their relationships. The distinction and the integration of basic types of knowledge and links in the same language are here essential. Depending on the type of knowledge or model, the instructional solutions will be different. For example, if the main knowledge to be learnt is a conceptual system, we might propose an induction scenario where examples and near examples are used to help the learner build precise concept definitions. On the other hand procedural knowledge might require the simulation of a process and its re-design, while the learning of a strategic approach might require study cases where principles are applied.

Consensus in Educational Science

In the MOT representation language, four basic types of knowledge units are used: facts, concepts, procedure and principles. There is a large consensus in educational science on this categorisation of knowledge. For example, the “Component Display Theory (CDT)” developed by (Merrill, 1994, p. 109-112) propose to distinguish also four types of « content »:

- a fact is an association between a date and an event or a name and a part;
- a concept is a set of objects, events or symbols sharing a number of common characteristics;
- a procedure is a set of steps enabling to conduct an action;
- a principle is a cause-effect relationship in a process.”

Other education researchers also propose to use four types of knowledge with slightly different meanings. For example, (Romiszowski, 1981, p. 242-243) also defines facts, procedures and principles as basic knowledge types and mentions that these are also the categories used by (Williams, 1977) to extend Bloom’s
and Gagné’ taxonomy of educational objectives, and previously by (Horn 1969) as a basis for his “information mappings” concept.

(Tennyson and Rasch, 1988, p. 372) develop an instructional design model focusing on « contextual (or strategic) knowledge” that they propose to be the subject of more learning activities than is usually the case in most learning environments. They distinguish three categories of knowledge:

• “declarative knowledge implies the learner’s attention to information in a domain and refer to « knowing that »;
• procedural knowledge implies “knowing how” to use concepts, rules et principles of a domain;
• contextual (or strategic) knowledge implies “knowing when and why” to use concepts, rules and principles of a domain.

Other authors such as (West, 1991, p 15-16) also group facts and concepts together as “declarative knowledge” and propose three categories:

• “declarative knowledge can be represented in memory by an associative network (Anderson, 1985) of proposition grouping facts and concepts; the networks can be semantic of episodic;
• procedural knowledge is know-how in the form of instructions to be executed in a certain order and at certain conditions;
• conditional knowledge amounts to know when and how to use a procedure with the following form: if these conditions are fulfilled, then do such and such thing”.

Foundation in Schema Theory

Despite these nuances, we can speak of a general consensus between educational researchers on four types of knowledge: facts and concepts (considered as declarative knowledge), procedures (or procedural knowledge) and principles (or contextual/conditional/strategic knowledge). This categorization is retained as the basis of the MOT graphic representation language.

All four types of knowledge are also considered in the framework of schema theory. The concept of schema is the essential idea behind the shift from behaviourism to cognitivism, the now dominant theory in psychology and other cognitive sciences. In the twenties, Jean Piaget (Inhelder et Piaget, 1958) uses concepts such as “schème”, “schéma”, “structure”, “stratégie” and “opération” to describe the cognitive processes involved in human learning. According to
Piaget, intellectual growth consists essentially in the development of schemas that are more and more structured logically and complex. In parallel to Piaget, “gestalt” psychologists such as Wertheimer (1945) had developed similar concepts of “entities”, “pattern” and “structure”. In the thirties, a British psychologist (Bartlett, 1932) had proposed that the interpretation of texts seems to be guided by pre-existing knowledge in the mind that he had called “schemas”.

Early in the sixties, mainly in USA, cognitive psychology and artificial intelligence start to develop. (Bruner, 1973) makes an essential contribution to demonstrate the psychological validity of knowledge representation and the construction of knowledge. (Newell and Simon, 1972) develop, on the same basis, a rule-based representation of the problem solving activity, while (Minski, 1975) defines the concept of “frame” as the essential element to understand perception, and also to reconcile the declarative and procedural views of knowledge.

Building on the convergence of these ideas, (Rumelhart et Ortony, 1977) have revisited the notion of schema and have proposed to use it in the following ways:

- schema as a data structure in the mind;
- schema to represent our knowledge about objects, situations, events and action sequence;
- schema as a scenario;
- schema as a theory to structure knowledge on a subject.

The distinction between conceptual and procedural schema has been accepted for a long time in cognitive science: the first ones structures data or objects properties while the others involves processes to transform and organize information. More recently a third category called “conditional or strategic schema” has been proposed (Paris et al., 1983). These schemas have a component that specifies the context and the conditions to trigger a set of actions or procedures, or to assign values to the attributes of a concept. These three categories correspond well to the consensus that has emerged in educational science.

Schemas play a central role in knowledge construction and learning. They guide perception, defined as an active, constructive and selective process. They support memorization skills seen as processes to search, retrieve or create appropriate schemas to store new knowledge. They make understanding possible by the comparison of existing schema with new information. Globally, through all these processes, learning is seen as a schema transformation enacted by
higher order processes. Learning is schema construction and reconstruction through interaction with the physical, personal or social world, instead of a simple transfer of information from one individual to another.

PRESENTATION OF THE MOT MODELING LANGUAGE

We will now present briefly the syntax and semantic of the MOT graphic modeling language, based on the notion of schema. For more details, see (Paquette 1996, 1999, 2003).

Schema help represent factual, conceptual, procedural or conditional knowledge graphically. We identify the main components of the schema, called attributes, and the type of value that these attributes can take: number, symbol, and other schemas. Depending if the schema is a concept, a procedure or a principle, the attributes will have different forms as shown in the simple example on figure 1.

![Figure 1](image)

There types of schemas.

**Conceptual schemas** can be described by simple or complex attributes. On figure 1, the attribute «Nb angles» takes a single integer value that cannot be decomposed further while «Type of figure» take another conceptual schema as value, the concept of a «Triangle». This schema could be decomposed by its attributes until we reach simple concepts. Also, some attributes can point to a conditional schema such as «is-rectangle » or to a procedural schema such as «Compute Area » that will provide a value, either boolean or numerical.
Procedural schemas aim to describe a set of actions that produce some output result. The attributes of such schemas are the different actions that compose the procedure. These actions can be other procedures such as “Verify if Is-rectangle” or principles such as “If Is-Rectangle, compute half-product of opposite sides to the right angle”. When an action is another procedure, another schema will represent it until we obtain a simple procedure such as “Obtain A, B, C”.

Conditional schemas can be composed of a set of principles or rules, each one composed of a condition part and a consequence part. When the condition is satisfied, the principle either imposes « integrity constraints » on a concept or trigger actions (procedures). This is the case here of the principle “Is-rectangle” that serves to test if a triangle has a right angle and triggers an action on its opposite sides.

Introducing the MOT Graphical Language Vocabulary

In the MOT graphic language we improve the readability and the user friendliness of graphs by externalizing the internal attributes of a schema with proper links to the encapsulating schema. For example, the link between the schemas “Triangle” and the “Rectangle Triangle”, implicit in figure 1 inside the schemas, will be shown explicitly on figure 2 using a is-a-specialization-of (S) link from the “Rectangle Triangle” schema to the “Triangle” schema.

Similarly, the links between the “Triangle” concept and its sides or angles are externalized using a composition (C) link. Note here that we use a composition link in a more general sense between a concept and any attributes that is also a concept, such as a triangle and its area. The link from an input concept to a procedure, and from a procedure to one of its results are both shown by an input/product (I/P) link. The sequencing between actions and/or conditions (principles) in a procedure will be represented by a precedence (P) link. Finally, the relation between a principle and a concept that it constrains, or between a principle and a procedure that it controls, will be represented by a regulation link (R).

Using these links, the simple example on figure 1 becomes the MOT model on figure 2 where relations between knowledge entities are more transparent and easy to interpret.

Syntax of the MOT Graphic Language

Concepts (or classes of objects), procedures (or classes of actions) and principles (or classes of statements, properties or rules) are the primitive objects
of the MOT graphical language. The type of the object is represented by geometrical figures as shown on figure 3, where each class or individual is represented by a name within the figure.

These objects are different types of schema whose attributes are all externalized explicitly and related to the schema using six kinds of links constrained by the following grammar rules:
1. All abstract knowledge entities (concepts, procedures, principles) can be related by an instantiation (I) link to a set of facts representing individuals called respectively examples, traces and statements.

2. All abstract knowledge entities (concepts, procedures, principles) can be specialized or generalized to other abstract knowledge using specialization S links.

3. All abstract knowledge entities (concepts, procedures, principles) can be decomposed, using C links into abstract knowledge of the same type, except procedures that can have principle components, and also principles that can have procedural components.

4. Procedures and principles can be sequenced together using P links.

5. Concept can be inputs to a procedure using an I/P link to the procedure, or they can be product of a procedure using an I/P link from the procedure.

6. Principles can regulate, using an R link, any procedure to provide an “external” control structure (such as a rule base in expert systems), to constrain a concept or a set of concepts by a relation between them (such as a definition or a law in a theory), or to regulate a set of other principles, for example to decide on conditions of their application.

Figure 4 represents these rules in the form of a metamodel.

Semantics of the MOT Graphic Language

Table 1 presents various possible semantic uses of these graphic symbols.
Types of Models

Using MOT objects and links, it is possible to construct increasingly complex systems of structured knowledge. For example, we can build representations equivalent to conceptual maps, flowcharts (iterative procedures) and decision trees, and also other types of models useful for educational modeling. Figure 5 presents five main categories of MOT models which are subdivided into subtypes. All these types of models have been used in a number of projects since the publication of the first MOT editor in 1999.

Of particular interest here are two of these types of models. The class "processes and methods" includes instructional design methods such as MISA, that we have totally described graphically using the MOT editor, but also, the learning scenarios, represented by multi-actor process graphs (collaborative systems).

The other interesting type of model is "laws and theories" where models are composed of concepts organized in specialization hierarchies, with principles defining their properties and relationships. Particular cases are ontology models that we will use to describe knowledge domains.

Modeling Formal Ontologies

Any type of knowledge representation, including text-based narratives or
informal graphic models, can be used to describe a domain of study. At the initial stage of design, the informal nature of representation is useful. The mind must be free to choose any representation that seems best suited for the educational project to be considered. Still, this very freedom does not necessarily facilitate the software processing of the representation.

Semi-formal modeling languages like MOT go part of the way in that direction unlike informal graphs built with any graphic editor such as Powerpoint. The MOT graphic syntax is structured and has a general unambiguous semantic. Using the MOT editor, models can be exported in many formats, including a native XML schema.

Using such a schema, software agents can perform different kind of processing, but still some ambiguity remains. For example, in a control
structure, texts within the principles that rule the procedures in a process have to be interpreted in non-ambiguous terms for the process to be performed. In fact, in instructional engineering applications, we had to constrain the MOT graphic language to enable the delivery of learning scenarios in a digitized platform like Explor@-2 (Paquette 2001). Even then, part of the transfer of the design to the delivery platform had to be done manually in order both to prevent enforcing unnatural graphic representations on the users or, at the opposite leaving ambiguities that prevent the software to execute.

After a phase where informal graphic design has cleared up ideas, we need to move from informal or semi-formal graphs to formal computable graphic representations. In this section, I will discuss formal knowledge domain representations. In the following one, I will address the representation of learning design processes.

**OWL-DL and Description Logic**

Knowledge in a subject domain can be represented in many ways: taxonomies, thesauri, topic maps, conceptual graphs and ontologies. We are looking for a formal standardized representation that would guarantee computability of the representation of possibly complex domains, not only in simple situations of life or highly constrained domains like mathematics, but general domains as found in the majority of educational situations.

Ontologies are good candidates for that purpose. They are composed of three kinds of entities:

- Classes (or concepts), grouping individuals, organized in “a-kind-of” specialization hierarchies;
- Properties (or role) between classes that describe binary relationships between them or from a class to a set of possible values;
- Axioms that state properties of these classes and relations (for example that some property is transitive or functional).

OWL, the Ontology Web Language is part of the growing set of World Wide Web consortium (W3C 2003) recommendations related to the Semantic Web. OWL provides three increasingly expressive sublanguages, each providing an XML schema definition to guaranty the compliance of a particular model to the standard form.

- “**OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints. OWL Lite provides a quick migration path for
thesauri and other taxonomies. Owl Lite also has a lower formal complexity than the other two representations.

- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.” (W3C 2003)

Each of these sublanguages is an extension of its simpler predecessor, enlarging what can be legally expressed and what can be validly deduced from the assertions in the ontology. Because of its foundation in description logics, and its computational completeness and decidability, OWL-DL provides an interesting framework to structure knowledge domain representation.

*Description Logics* (Baader et al 2002) are important knowledge representation formalisms unifying and giving a logical basis to the well known traditions of Frame-based systems, Semantic Networks and KL-ONE-like languages, Object-Oriented representations, Semantic data models, and Type systems.

A description logic is a subset of first-order predicate logic that consists of

- A set of unary predicate symbols used to denote concept (or class) names
- A set of binary relations used to denote role (or properties) names
- A recursive definition for defining new concept terms from concept names and role names using constructors and axiom restrictions.

Common constructors or restrictions are intersection or conjunction of concepts, union or disjunction of concepts, negation or complement of concepts, value, universal and existential restrictions, enumeration of individuals, inverse relationship between properties, equivalence of classes and properties, transitivity or symmetry of properties, etc.
MOT+OWL

OWL-DL provides a precise XML schema for each component of an ontology but no graphic representation per se. Some ontology editors like HOZO or PROTEGE provide partial graphical views of the ontology, but the construction of a model is essentially form-based.

This is why MOT+OWL was built: to provide a complete formal graphic representation of all OWL-DL components that can combine the virtue of user-friendly graphical construction with the computational capabilities of a formal specification. In the context of the MOT representation system, ontologies, in particular OWL-DL constructs, correspond to a category of models called theories. They can thus in principle be modeled graphically using the MOT syntax. We have thus specialized our MOT+ language and graphic editor. The specialized MOT+OWL editor uses adaptation of the objects and links available in the MOT+ editor to cover all the OWL primitives.

Figure 6 shows the complete set of graphic symbols in MOT+OWL. Three kinds of objects are used to represent classes (rectangles), properties (hexagons) and individuals (rectangles with cut corners). In the general MOT+ language, these correspond respectively to concepts, relational properties and facts.

The upper part of the figure presents the different constructors or axiom restrictions between classes, between classes and individuals and between individuals. Here some links from the general MOT+ language are used such as the specialization (S) link for sub-class/class relationship or the instantiation (I) link for class/individual relationship. New links are introduced for equivalent, disjoint and complementary classes, as well as for identical and different individuals. Finally labels are used on class symbols to denote more complex constructions or restrictions. For example, to construct the intersection of two classes, it is not sufficient to assert that the intersection class is a subset of the super-classes. Here the label is essential to denote that Class3 is exactly the intersection of the super-classes.

The lower part of figure 6 completes the graphic language by presenting properties, their relationships and restrictions. The basic link here is the MOT regulation (R) link. The first property group of three objects is read “The domain class is in relation by the property with the range class”, while the second one is read “The domain class has property values of a certain data-type”; in that last case, an item in the OWL set of data-types can be selected in a menu of the MOT+OWL editor. The two next property groups use labels to represent universal or existential restriction on the value of a property. The other relations between properties or labels on properties have straightforward interpretations.
Let us note here that relations between properties can be represented graphically because the properties have been represented as objects. In ontology editors like PROTÉGÉ or HOZO, the property names are put on links between classes. The consequence is that only part of the ontology is made visual, the relations between properties are asserted in table forms.

FIGURE 6
A simple task ontology for multi-actor scenarios.

Let us note here that relations between properties can be represented graphically because the properties have been represented as objects. In ontology editors like PROTÉGÉ or HOZO, the property names are put on links between classes. The consequence is that only part of the ontology is made visual, the relations between properties are asserted in table forms.
Translation to OWL-DL

Any MOT+OWL local model correspond to an assertion in the DL subset of First-order logic, as well as to a fragment of a corresponding OWL-DL XML schema. Table 2 provides some examples of these correspondences.

<table>
<thead>
<tr>
<th>MOT+OWL and 1st Order Logic</th>
<th>OWL-DL XML Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Intersection of classes</td>
<td><code>owl:intersectionOf rdf:parseType=&quot;Collection&quot;&gt;</code></td>
</tr>
<tr>
<td><code>(\forall x) (Class3(x) = (Class1(x) \land Class2(x)))</code></td>
<td>List of super-class descriptions</td>
</tr>
<tr>
<td><code>owl:intersectionOf&gt;</code></td>
<td><code>owl:Class&gt;</code></td>
</tr>
<tr>
<td><code>&lt;owl:intersectionOf&gt;</code></td>
<td><code>&lt;/owl:intersectionOf&gt;</code></td>
</tr>
<tr>
<td><code>&lt;/owl:Class&gt;</code></td>
<td><code>&lt;owl:Class&gt;</code></td>
</tr>
<tr>
<td>2) Equivalent classes</td>
<td><code>owl:Class rdf:about=&quot;#name_class1&quot;&gt;</code></td>
</tr>
<tr>
<td><code>(\forall x) (Class1(x) = Class2(x))</code></td>
<td><code>&lt;equivalentClass rdf:resource=&quot;#name_class2&quot;/&gt;</code></td>
</tr>
<tr>
<td><code>owl:Class&gt;</code></td>
<td><code>&lt;/owl:Class&gt;</code></td>
</tr>
<tr>
<td>3) Definition by extension</td>
<td><code>owl:Class&gt;</code></td>
</tr>
</tbody>
</table>
|    `(\forall x) (Class(x) = ((x = I_1) \lor (x = I_2) \lor ... \lor (x = I_n)))` | `<owl:oneOf rdf:parseType="Collection">
|    `owl:Class>` | `owl:Thing rdf:about="#name_individual1">` |
|    `<owl:oneOf>` | `owl:Thing rdf:about="#name_individual2">` |
|    `<owl:oneOf>` | `owl:Thing rdf:about="#name_individualN">` |
|    `owl:Class>` | `</owl:oneOf>` |
| 4) `(\forall x)(\forall y)(\forall z) ((P(x,y) \land P(x,z)) \supset y=z)` | `<owl:FunctionalProperty rdf:about="#name_property" />` |
| 5) `(\forall x)(\forall y) (P1(x,y) = P2(y,x))` | `<owl:ObjectProperty rdf:ID="name_Property1">` |
| 6) `(\forall x) (Domain(x) = (\exists y) (P(x,y) \land Range(y)))` | `<owl:Restriction>` |
| 7) `(\forall x) (Domain(x) = (\exists y) (P(x,y) \land Range(y)))` | `<owl:Restriction>` |
|    `owl:Restriction>` | `<owl:Restriction>` |
|    `<owl:onProperty rdf:resource="#name_property" />` | `<owl:cardinality rdf:datatype="&xsd;nonNegativeInteger" >3` |
|    `<owl:cardinality>` | `</owl:cardinality>` |
|    `<owl:Restriction>` | `</owl:Restriction>` |

TABLE 2
Correspondence between some MOT+OWL Graphic Elements and XML Fragments.
• The first graph declares that Class3 is a subclass of both Class1 and Class2, using the MOT standard “S” link; the little icon on the Class3 object declares that further more, Class3 is the intersection of the two other classes.
• The second one declares the classes are equivalent, that they have the same extension of individuals.
• The third graph, defines a class by enumeration, stating that the Class1 is composed of a list of individuals and only of those individuals.
• In the fourth graph, the label on the property object asserts that this property is a functional relation.
• In the fifth graph, a property is defined as the inverse of another one.
• The sixth graph defines a certain class “Domain” as a restriction composed of those individuals who have at least one of their property values in the range: for example, the subclass of persons who “have diploma” value that is a “university degree”.
• Finally, the last graph states a cardinality axiom: any object in the domain of a property has at least 3 corresponding property values in the range.

A Simple MOT+OWL Model

We present here, as an example, a simple ontology aiming to identify agriculture practices that influence the greenhouse effect. This ontology could serve in a learning environment where students have to find out agriculture alternative practices, in at least five agriculture domains, and to build a transition plan towards the replacement of the old practices by the more ecological ones. The ontology would serve in a browsing mode to access related resources (e.g. annotated by the ontology) and to launch search agents to find persons, information resources and learning activities useful to achieve the learning activities.

Figure 7 presents a simple example to illustrate the use of the MOT+OWL editor. The upper part of the graph presents the top levels of three hierarchies of concepts linked by sub-class links “S”: agricultural practices, fertilizers and gases. Some properties of these concepts are shown on the graph. An Agriculture practice, such as Rice Production Processes, has inputs including fertilizers and outputs that can be gases. Fertilizers can also produce gases, some of which are greenhouse gases. Figure 7 also shows a few of the individuals (or class instances) that will constitute the knowledge base. Here we see an agriculture practice, Technical Rice Production, having among its outputs methane gas. It also has Nitric Oxide amongst its inputs, a chemical fertilizer that produces Carbon Dioxide. Both of these gases are example of greenhouses gases harmful to the environment.
We could now develop this ontology further into sub-graphs to cover the domain more completely and use it to reference the documents and activities in a learning scenario. For this, sub-graphs of figure 7 would be associated to actors, activities and resources used and produced by the activities, to represent their knowledge.

An Ontology for IMS-LD multi-actor scenarios

With ontologies, we can in principle describe formally any informal model of the MOT taxonomy of models presented on figure 5, with the restriction that they are amenable to a representation in the description logic at the basis of OWL-DL. It is obvious that ontologies can represent most conceptual models or laws and theory models. It is less evident that they can also describe procedural models (task ontologies). Procedural and process/methods models are important for our purpose because learning designs for courses require the definition of multi-actor processes. We will present here one such model.
The IMS-LD standard specification defines an XML schema for models of multi-actor scenarios or designs built with a number of components. The central components are procedural knowledge elements or task: a method is decomposed into plays, plays into acts, acts into a number of role parts where a role is performed towards an activity. Learning or support activities can be grouped in any level of activity structures. Activities use and produce learning objects and services within a local environment.

Figure 8 presents a MOT+OWL graph that corresponds exactly to the main conceptual structure of a learning design presented in the IMS-LD information model (IMS-LD 2003, p. 10). On the figure, C properties represent the “is-composed-of” relationship which has a similar meaning as the C link in standard MOT models, or the aggregation link in UML models.

This example illustrates how functional relations between components of multi-actor processes such as the method part of a learning design can be
represented by ontologies. Such ontologies can be used to test, for example, the conformance of a specific learning design model to the IMD-LD XML schema (Amorim et al., 2006) and to execute that learning design in the context of ontology-driven system engineering (Magnan and Paquette, 2006). We will address this important question next.

**ONTOLOGIES AND EXECUTABLE LEARNING DESIGNS**

Even though ontologies are theoretically sufficient to describe multi-actor processes and learning designs formally and computationally, we need to take into consideration usability, implementation and deployment issues. In other words, we need alternative representations of task ontologies such as IMS-LD that are not only formal but also transparent to user and useful to support the design and delivery of learning environments.

**Learning Designs in MOT+LD**

Such representations exist in workflows models such as BPMN, the Business Process Modeling Notation (OMG, 2005), in some instructional design graphic software such as LAMS (Dalziel, 2005) and in our own MISA scenarios using the standard MOT editor (Paquette 2003). Unfortunately, these representations are either informal like LAMS or semi-formal like MOT, or they are incomplete for learning design modeling, such as BPMN workflow models that place all the emphasis on the flow of control and not on the resources or the knowledge used or produced during the learning delivery process.

To address this problem, we have developed another MOT specialization, MOT+LD. Figure 9 shows all of the graphic symbols used to model IMS-LD scenario models according to the IMS-LD ontology or information model (partly shown on figure 8). It shows a rather straightforward use of the composition link (C link) for environments, roles and method components. The use of input/product (I/P-link) and precedence (P-link) links is also clear and unambiguous but show some restrictions on procedure sub-types. Finally, the regulation (R-link) link is used to associate learner and staff roles to environments or activity structures, learning or support activity, or to associate time limits or on-completion messages to some of the actions except the method; it is also used to associate a completion rule to an action, or to provide the number to select in an activity structure when options are proposed.¹

¹ A more detailed discussion has been published in (Paquette et al., 2006).
Technically, the native MOT XML schema was extended to cover the symbols on figure 8 and to parse it into a valid IMS-LD XML schema, which correspond to the ontology presented on figure 9. A post-validation mechanism was built into the parser informing the designer whether an IMS-LD rule had been violated and where to find it in the model. Finally, all the IMS-LD (See IMS 2003 Best Practices) examples were modelled and tested, including the complex Versailles collaborative role playing example. A test was made by uploading them into the RELOAD editor (RELOAD 2005) a form-based LD editor. This exercise resulted in very small discrepancies between our analysis of the specification and minor corrections were made to the MOT+LD editor or parser to produce the present version.

**Running Learning Designs**

Many examples of learning designs have been produced by different groups using this editor\(^2\). Figure 10 shows part of a simple example of a learning unit

\(^2\)A version of the MOT+LD editor and these examples are available at the IDLD project Web site www.idld.org.
on solar astronomy presented recently at a workshop (Paquette and Léonard, 2006). We see from this model of act 2.0 entitled “Discussion”, that an act and its learning and support activities are represented as MOT procedures. So are method, plays and acts in other parts of the model. The kind or sub-type of each procedure is indicated by little label at the right lower corner below the ovals representing the procedures.

Similarly, roles such as “Teacher”, “Learners team A” and “Learners team B” are represented by different kinds of MOT principles. Environments, learning objects, services and outcomes are represented by different kinds of MOT concepts. In this case, standard MOT links are used and only C, P, R and I/P links are sufficient to cover all the components of an IMS-LD learning design.

Once a learning design is produced in MOT+LD and exported in a standard XML file, it can be read and executed by any IMS-LD compliant eLearning system like RELOAD-player (2005), and COPPERCORE (2005). Another way is to use an ontology-driven system such as the TELOS scenario editor we are building (Magnan and Paquette, 2006). Basically, it implies to embed the IMS-LD ontology into the general TELOS ontology and execute in through queries to the knowledge base in a manner similar to the logic programming paradigm.
Semantic Annotation of Learning Design Components

In (Paquette and Marino 2005) we have discussed briefly the strengths and weaknesses of the IMS-LD educational modeling specification. IMS-LD remains the first pedagogically-oriented standardized specification that implements a complete task ontology for learning designs accepted internationally, but it is weak on knowledge representation of the actors, activities and resources that compose a learning design.

Actually, in IMS-LD, the only way to describe the knowledge in the activities or in the resources is to assign optional educational objectives and prerequisites, to the unit of learning as a whole and/or to all or some of the learning and support activities. Objectives and prerequisites correspond to entry and target competencies. They are essentially unstructured pieces of text composed according to the IMS RDCEO specification (IMS 2002).

Unstructured texts are difficult to compare. Consistency checking between different levels of the LD structure cannot be supported computationally. Even at the same level of a learning design, for example within an act, no relations exist between the knowledge processed in learning activities and the knowledge present in input or outcome resources, or the actors’ knowledge and competencies. In fact, in IMS-LD the knowledge represented in learning resources is not described at all, and the actor’s knowledge and competencies are only indirectly defined by their participation in learning units or activities, as long as educational objectives have been associated to the activities.

What we need is first a qualitative structural representation of knowledge and competencies involved in activities, resources and actor’s roles. This can be done using domain ontologies such as the one presented above for the domain of eco-agriculture. As a first step, we can use the MOT+ editor to put side by side a learning design using the MOT+LD module, and a domain knowledge ontology using the MOT+OWL module. An example is shown on figure 11. The left hand window is the learning design on figure 10 for a learning unit on the subject of the solar system. The right hand window presents part of a simple domain ontology for the solar system.

We define a semantic annotation as simply a mapping from the ontology to the learning design that associates knowledge elements to components of the learning design. On the figure, we see that data on the orbital period of planets in the solar system has been associated to a learning object in the design, which in this case is a powerpoint presenting this data to team A. This resource is an input to learning activity 2.1.A, but it is not the only input to this activity. There is also another resource (clues A) that gives additional information to team A. There is also a chat between team A members that will bring additional information to
each participant. As a result, the sub-model of the ontology associated to activity 2.1A should logically correspond to the union of the sub-models of all input resources to the activity. Finally, the figure shows that most of the ontology model should be the subject of the discussion, since there is another team, team B that has more information to bring to the discussion using also information from input resources and in a team B chat.

This example shows how semantic annotation can help guide the construction of learning designs or evaluate their coherence. By associating the right amount of knowledge to the different resources and activities, a designer can build a coherent design that will trigger collaboration between learners, or help a trainer decide on its intervention, or guide the actions of an intelligent tutoring system, and, in general support the assessment (informal or formal) and the evolution of the learners’ competencies.

**Knowledge and Competency Annotations**

Associating knowledge to components of a learning design is essential but not sufficient. To say that a person possess some knowledge is not enough. Is that person able to give examples of that knowledge? Is she able to apply it, to add elements to it, or to evaluate it? If we use only knowledge annotations from
an ontology without stating the mastery level of that knowledge, we limit ourselves to a coarse granulation of sense and, as a consequence, to weak support services to the learner. The evolution of a learner on a competence scale materializes a learning process: therefore, it should be managed explicitly and expressively.

In other words, we need measures of knowledge mastery, a weighted ability defined on that knowledge that corresponds to the concept of competency related to knowledge. A thorough discussion of generic skills, competency and performance exceeds the scope of this text. For this the reader is referred to previous publications (Paquette 1999, 2002, 2003). The generic skill taxonomy has been defined by combining elements of an artificial intelligence taxonomy (Pitrat 1990), a software engineering taxonomy (Breuker and Van de Velde, 1994; Scheiber et al. 1993) and two educational taxonomies (Bloom 1975; Romiszowski 1981). Here, we will only present an example to show how semantic annotation can be extended to competency annotations and to quantitative annotation using any generic skill taxonomy.

The Explor@-2 delivery system (Paquette 2001) has been based from its inception on two structures, the instructional structure, corresponding to the learning design, and the knowledge/competency structure, corresponding to a simplified domain ontology (in fact a thesaurus of terms) augmented with attachments from a competency/performance ontology. This second ontology provides a two dimensional scale on which to situate knowledge mastery of learners at any time. This quantitative measure of competency enables the definition of a competency gap that the learning environment is designed to address.

Figure 12 shows screens of a simple knowledge/competency editor provided to Explor@-2 designers. Compare to the model for eco-agriculture on figure 6, the ontology is simplified into a hierarchy of concepts, to which competencies are further associated using the knowledge/competency editor on the right side of the window. Another hierarchy, the activity structure, corresponds to the learning design. It is built using the activity editor shown on the external window of figure 12.

In the knowledge/skill editor, the terminal nodes are skills selected in a taxonomy of generic skills that are associated to their parent nodes. If we select one of the skills (on the figure, “Analyze-6” is selected), plus some performance criteria (selected in the little window on the left), we can associate to its parent knowledge, here “Agriculture Processes”, a target competency and an entry competency. These are shown on the right side of the knowledge/ skill editor. This editor plays a similar role than the ontology editor, extended with entry and target competencies attached to the ontology classes or concepts.
Competency statements are texts that are used mainly for display purposes but they have here a precise interpretation as knowledge (skill + performance) couples. For example, the competency statement “classify the agriculture processes according to their greenhouse effect, autonomously in difficult situations”, is interpreted as analyze (skill level 6), autonomously in difficult situations (performance B), applied to agriculture practices (knowledge). This is the target competency expected from persons involved in some formal or informal training on the subject.

The generic skills scale and the performance levels are two ordered sets of values that enable comparison between competency statements for the same knowledge element. Skill are ordered from 1 to 10 and performance levels A, B, C and D, estimated from a combination of increasingly demanding performance criteria, can be transformed in numbers (A = 2, B = 4, C = 6 and D = 8). We then obtain a metric enabling to represent the distance between entry and target competency for that knowledge (agriculture practice). On the figure, a designer has evaluated that the entry competency is “Apply-5” at a performance level A, which means that learners will have to move up to the “Analyze-6” level, with an expected
increase of performance from A to B. In numerical terms, learners are expected to move from 52 to 64 on a 100% scale, closing a relative competency gap of 12%.

On figure 12, the lower right corner of the Knowledge/Skill editor (more precisely the “Add” button) is where the designer associates the selected knowledge/skill + performance couples (and the entry and target competency statements) to the components of the activity structure shown in the activity editor. In this way, all the activities and the resources (placed on terminal nodes of the activity structure) can be annotated with knowledge and competencies. This example shows one way to extend a knowledge annotation to a competency annotation of resources in a learning design.

Technically, in order to annotate resources according to their competencies in a general way, we have proposed (Rogozan and Paquette, 2005) an RDF-based method that uses two ontologies: a knowledge domain ontology specifying a consensual view of a subject-matter (Breuker et al. 1999) and a skill/performance ontology specifying the generic mastery levels that may be applied to any knowledge element from a learning domain (Paquette 2003). The skill/performance ontology has two root classes: Generic Skill and Performance. The Performance sub-classes refers to the Performance degree of a generic skill when it is applied on a knowledge element.

These ontologies provide precise structural meaning and relative quantitative measures within an ordering of mastery levels on each knowledge element. Normally, the learner will evolve from an entry competency level to a target competency level for each major knowledge element in the domain ontology. At any time during learning, his/her actual competency can be evaluated by a trainer or self-evaluated by the learner himself.

The competency annotations of the activities, the resources and the actors in a Learning Design can be used in many ways to support the design and delivery of learning environments.

• At design time, they can help designers (or learners acting as their own designers) to:
  ◦ prepare a sequence of learning units that should increase progressively the mastery level of learners,
  ◦ identify learning resource (documents, tools, activities, persons) to be included in a learning design that possess the right knowledge and the right mastery level to help learner progress,
  ◦ provide criteria to form teams with learner having homogenous or heterogeneous mastery levels, or to plan different paths or plays for learners with weak or strong mastery patterns,
• At delivery time, they can help learners and trainers to:
  • evaluate the progress of learners’ competencies for important knowledge elements,
  • detect learners at risk by comparing their evolution pattern to the group average and alert learners, trainers and designers on possible flaws in the learning environment,
  • find appropriate resources or units of learning in a learning object repository where resources have been referenced with semantic annotations,
  • build /maintain user models to guide trainers’ intervention, to trigger an intelligent advisor or a tutoring system, or to add information to an e-portfolio system.

EVALUATION OF THE KNOWLEDGE REPRESENTATION PARADIGM

I now conclude with a discussion of the knowledge representation paradigm developed in these pages. First of all, let us agree that there is no universal representation that is best for all things. The best representation is the one that fits the purpose for which it is used.

Here, our purpose it to improve the design and delivery of computer-supported learning environment. It is not to find the most economical language built with the fewest primitives possible, or one that serve to illustrate elegant and profound theories, or one that best automates learning and teaching, or one that works best for the computer scientist or is the most efficient in rigidly structured or highly specialized domains.

Our goal is to support the design and delivery of real learning and knowledge intensive environments that are technology-enhanced.

Computer-supported learning environments are generally built by designers, used by learners and educators who are not in general computer scientists. These actors play different roles towards learning and prefer a variety of leaning/teaching strategies that must all be supported. They might be involved in K-12, university, workplace or continuing education. They are interested in a large variety of knowledge domains, are seeking to acquire or help others acquire a large spectrum of cognitive, socio-affective or psychomotor skills.

For these reasons, I assert that the knowledge representation we use should be graphic, user-friendly, general, declarative, standardized and computable. Let me discuss these criteria one by one.
Graphic

The benefits of graphical cognitive modelling have been eloquently summarized by many authors, Ausubel (1968), Dansereau (1978) and Jonassen (1993) to name a few. Graphical models illustrate relationships among components of complex phenomena. They uncover the complexity of actors’ interactions. They facilitate the communication about the reality studied. They favour the global comprehension of studied phenomena. They help grasp the structure of related ideas by minimizing the use of ambiguous natural language texts.

As an example, entity-relations graphs reduce ambiguity compared to a natural language description, but some ambiguity remains on the interpretation of the terms written on the links or on the nodes. Ambiguity can be reduced further by the use of standardized typed objects and typed links. It can be completely removed by the use of formal language such as ontologies, for which we have produced a completely graphic language, an editor and computer agents that can process the model for a variety of purpose.

User-friendliness

Not all graphic modeling languages are user-friendly. A good counter-example is UML. The large number models and symbols involved require considerable expertise for interpretation and for the construction of a system’s architecture. Furthermore, each type of model captures a different viewpoint on the information and it is useful to mix them in the same graph to provide a global view of a subject domain.

In our view, a representation system intended to support pedagogy must be easy to use without technical or scientific mastery after a relatively short period of initiation. Dansereau and Holley, (1982) have studied experimentally the use of different sets of graphic symbols by learners. Their results show that typed links are preferred by the majority of learner, as long as there are not two few nor two many links. In other words, the components of a graphic system must be easy to interpret. The meaning of links between knowledge elements must be sufficiently distinct from one another, yet they must capture natural ways of thinking. The number and variety of MOT applications in the last fifteen years qualify on that account.

Generality

Generality means that the representation language should have the capacity to represent, with a relatively small number of objects and link categories, all knowledge in very different subject domain, at various levels of granularity and formality.
It should, for example, enable the representation of simple models such as a multiplication table, up to complex models such as multi-actor workflows, rule-based knowledge system, methods and theories. It should also embed equivalent representations to commonly used graphs such as conceptual maps, semantic networks, flowcharts, decision trees or cause/effect diagrams.

The graphic language should be scalable from informal graphs, up to semi-formal and totally unambiguous formal models. At the informal level, the modeling user should be freed from too many constraints that limit the pace of thought generation and structuring. Later on, as the model becomes closer to a system, more formality is required to eliminate ambiguities. An integrated representation framework like MOT facilitates thought organization and communication between humans at the early stages of knowledge elicitation where the process is more important than the result. Later on, when the model is more important than the process, the graphic language should make it possible to integrate constrained elements to produced totally unambiguous descriptions that can be exported to set of symbols, such as an ontology, to be processed by computer agents.

**Declarative**

Graphic language can be procedural or declarative. There have been many discussions in the past on that issue. Procedural graphic languages have been built in the past; essentially extending flowcharts to promote graphical programming that would produce code directly. Our proposal is to use, as much as possible, a declarative graphic language, for a number of reasons.

Firstly, it is easier for a person to declare the components of his/her knowledge than to describe also the way it should be processed. Procedural languages mix up the knowledge with the processing on the knowledge, blurring the interpretation of the knowledge representation. In expert systems for example, the execution instructions are not wired-in the program, but externalized and made visible in a knowledge base on which a general inference engine proceeds.

Secondly, the same model, if declarative, can be used more easily for different applications, not necessarily the one for which the processing has been planned in a procedural program. For example, rules for operating or for diagnosing a component-based system can be applied to different conceptual models describing a car, a software system or a learning environment. This can be done by querying the model using an inference engine, in a Prolog-like manner. Ontology-driven architecture, a promising approach, is based on that very idea.
Thirdly, the processing knowledge itself can be given declaratively, so that higher order meta-knowledge, can be also singled-out. This idea is similar to structural analysis (Scandura 1973) and it is exactly the way we should see the relation between generic skills and domain knowledge in a competency. A generic skill, like apply, synthesize or evaluate is meta-knowledge applied to domain knowledge, stating for example that a certain concept can be applied, synthesized or evaluated by somebody. Many generic skills have been described by MOT process-model, thus providing skeletons of activities where the structure of the skill can guide its application to specific domain knowledge (Paquette 1999). This is similar to the way the KADS methodology propose to operate when task and inference models are applied to domain knowledge (Breuker and Van de Velde, 1994)

**Standardized**

Standardization is an important property to enlarge knowledge use and communication between users, persons or software agents. At the informal level, each model constructed by a person must be interpretable by another person. At the formal level, the communication capabilities extend to software agents. This is why we have standardized the MOT representation system as a tool within our research organization, facilitating the communication between participants in sometimes very different projects. The move towards graphic versions of standards like IMS-LD for learning designs and OWL-DL for ontologies adds wider communication capabilities between researchers and educators while at the same time adding formal non-ambiguous interpretation for machine processing.

**Computable**

Computability is a step ahead from standardization. It means not only that the graphic model can receive a non-ambiguous formal representation that can be processed by computer agents (this is the case of MOT+LD models), but that this formal representation is complete, that is all conclusions are guaranteed to be computable, and decidable, that is all computations will finish in finite time. These considerations have motivated the construction of the MOT+OWL graphic language as an equivalent to the OWL-DL language based on description logic while much simpler and user friendly than other editors we are aware of. Base on this standard, the use of models built with MOT+OWL will benefit from the growing number of tools and applications that are and will be developed by the international community.
CONCLUSION

In the last ten years, our main goal has been to synthesize and combine various forms of graphical representations that are useful for educational modeling, using an integrated graphical formalism. In (Paquette 2003), we have shown that very different kinds of representation, conceptual maps, flowcharts, decision trees and others, can all be modeled more precisely, using the MOT graphical language based on typed objects (concept, procedures, principles, facts) as well as few typed links.

With this set of primitive graphic symbols, it has been possible to build very different graphic models, from simple taxonomies to ontologies, more or less complex learning designs, delivery process, decision systems, methods etc.

Besides its generality, the MOT graphical representational language has been proven sufficiently simple and friendly to be used by persons with non-technical background in many different contexts either at an informal level, as a graphical “think pad”, at a semi-formal level where the models are precise enough to design learning environments, or at a formal level where ontologies and learning design models are produced totally graphically and declaratively with no ambiguity, thus enabling direct processing by computer agents or systems.

The team involved in the construction of TELOS$^3$ is in the process of building a system to assemble learning environments guided by the semantic annotation of resources. This system is ontology-driven which means that its blueprint is defined declaratively as a technical ontology, and its execution will proceed by requests to the ontology (Paquette, Rosca et al 2006, Magnan and Paquette 2006). The challenge here is to reduce the need for the traditional trade-off between power and simplicity, two conditions that should be present for better computer-supported educational environments.

REFERENCES


The TEleLearning Operating System is an assembly workbench to build on-line learning and knowledge management platforms, developed in the Canadianaian LORNET research network, led by the author.
Novak and Gowin, 1985).


