# Building Graphical Knowledge Representation Languages - From Informal to Interoperable Executable Models

Gilbert Paquette
*CICE Research Chair, LICEF Research Center, Télé-université/UQAM, Montréal, Canada*
gilbert.paquette@licef.teluq.uqam.ca

**Abstract**

*We summarize the evolution of a Graphic Knowledge representation language developed at our research center that has served in many applications throughout the years and, more recently to help develop the TELOS system. We first underline the theoretical research on which the language is built. Then we address the question of standardizing a knowledge modeling tool, first designed as an informal thinking aid. One standard specialization of the language is embedded in a tool to produce OWL models totally graphically. Another one enables users to produce standard IMS-LD multi-actor learning scenarios or workflows. The most recent evolution is the TELOS function editor that generalizes both IMS-LD models and business workflow models. We conclude by discussing the properties of the representation language that have benn found most useful.*

## 1.1. Introduction

The graphic representation formalism that we present here [1, 2] has been tested for the last 10 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 instructional methods or research projects processes.

In the first introductory section we discuss the basis for the graphic knowledge representation language. In the second one we present the main characteristics the MOT[1] language. Even at this informal level, the language constitutes a useful tool for precise definition and communication. In the third section we present a specialization of the graphic language to MOT+OWL, to represent domain knowledge and competencies as ontologies, thus bringing the representation language at a formal and computational level. In the fourth and fifth section, we address another specialization, first to enable building standard learning designs, and its generalization to a functional aggregation editor, a central core component of the Telelearning Operating System (TELOS) developed by the LORNET research network.[2]

## 1.2. 1. Basis for a Graphical Knowledge Representation Language

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. *Flowcharts* serve as graphical representations of procedural knowledge or algorithms. *Decision trees* are another form of representation used in various fields, particularly in decision-making or expert systems.

All these representation methods are useful at an informal level, as thinking aids and tools for the communication of ideas, but they have limitations. One is the imprecise meaning of the links in the model. Non typed arrows can mean many things, sometimes within the same graph. Another one is the ambiguity around the type of entities. Objects, actions on objects and statements of properties about them are all mixed-up, which make graph interpretation a fuzzy and risky business.

Another difficulty is 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 maps as 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 [3], Conceptual Graphs [4], Object modelling technique (OMT) [5], KADS [6], or the Unified Modeling Language (UML) [7]. These representation systems have been built for the analysis

[1] This acronym means “Modeling using Object Types”

[2] See the LORNET Web site at www.lornet.org

and architectural design of complex information systems. The most recent ones require the use of up to eight different kinds of model so the links between them become rapidly hard to follow without considerable expertise.

Our initial goals were different. We needed a graphic representation system that was 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 computer-based educational environments and their relationships.

There is a consensus in educational science to distinguish four basic types of knowledge entities (facts, concepts, procedure and principles), despite some diversity on the terminology and definitions. See for example, the work of Merrill [8], Romiszowski [9], Tennyson [10], and West [11]. 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, a now dominant theory in psychology and other cognitive sciences, based on the pioneering ideas of Piaget [12] and Bruner [13].

In the early seventies, Newell and Simon [14] had developed, on the same basis, a rule-based representation of the human problem solving activity, while Minski [15] had defined the concept of "frame" as the essential element to understand perception, and also to reconcile the declarative and procedural views of knowledge.

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 seen as 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.

The distinction between conceptual and procedural schema has been accepted for a long time in cognitive science. More recently, a third category called "conditional or strategic schema" has been proposed [6]. These schemas have a component that specifies the context and the conditions to trigger a set of action or procedures, or to assign values to the attributes of a concept. These categories map very well on the existing consensus in educational science.

## 1.3. 2. The MOT Graphic Knowledge Editor

We will now present briefly the syntax and semantic of the MOT graphic modeling language, based on the notion of schema. Here, we could use graphs similar to UML object models to represent the attributes that describe a schema with different formats according to their type. In the MOT graphic language [1, 17, 18], we try to improve the readability and the user-friendliness of graphs by externalizing the internal attributes of a schema into other schemas, with proper links to the original schema. For example, the link between the schemas “Triangle” and the “Rectangle Triangle” is shown explicitly using a specialization (S) link from the later to the former concept. Links between the “Triangle” concept and its sides or angles attributes is externalized using a composition (C) link. The links from an input concept to a procedure and from a procedure to one of its products are both shown by an input/product (IP) link. The sequencing between actions (procedures) and/or conditions (principles) in a procedure is 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, this simple example on triangle concepts becomes the MOT model on figure 1 where relations between knowledge entities are transparent, mixing the types of entities and links.

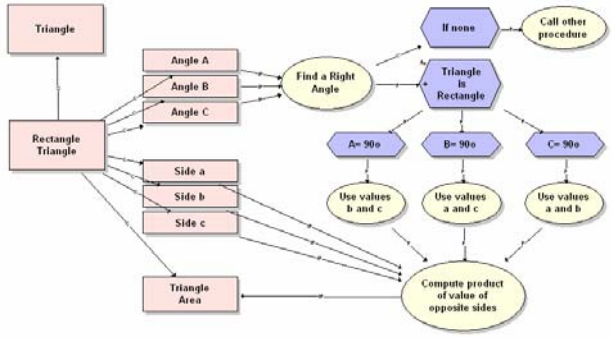



Figure 1 – A simple MOT model

*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 2, where each class or individual is represented by a name within the figure.

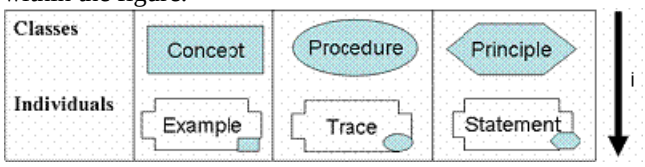



Figure 2 – Types of knowledge units in MOT

These objects are different types of schema whose attributes are all externalized explicitly and related to the schema using six kinds of typed links constrained by the following grammar rules:

1. All abstract knowledge entities or classes (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 other entities, generally of the same type.
4. Procedures and principles can be sequenced together using **P** links.
5. Concept can be inputs to a procedure using an **IP** link to the procedure, or products of a procedure using an IP link from the procedure.
6. Principles can regulate, using an **R** link, any procedure to provide an "external" control structure, to constrain a concept or a set of concept by a relation between them, or to regulate a set of other principles, for example to decide on conditions of their application.

Figure 3 summarizes these grammar rules of the MOT graphic language in the form of an abstracted graph where the nodes represent types of MOT objects and the arrows are valid link between them.

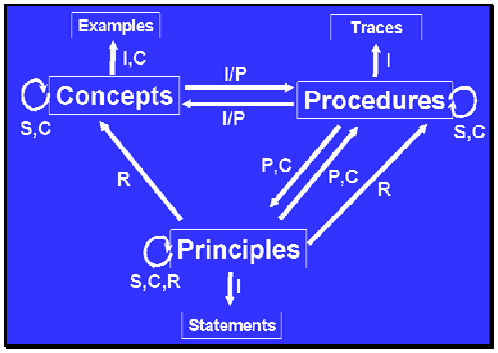



Figure 3 – The MOT metamodel

There are various possible semantic interpretations of these graphic symbols. *Concepts* can be object classes (country, clothing, vehicles…), types of documents (forms, booklets, images), tool categories: (text editors, televisions…), groups of people (doctors, Europeans…), event classes (floods, conferences, …). *Procedures* can be generic operations (add numbers, assemble an engine…), tasks categories (complete a report, supervise aproduction…), activities (take an exam, teach a course,…), instructions (follow a recipe, assemble a device…), or scenarios (of a film, of a meeting, of a learning module. *Principles* can state properties of objects (cars have four wheels), constraints on procedures (the tasks must be completed within 20 days), cause/effect relationships (if it rains more than 25 days, the crop will be in jeopardy), laws (any metal sufficiently heated will stretch out), theories (the laws of the market economy); rules of decision (advising on an investment), prescriptions (medicinal treatment, instructional design principles), etc.

With this set of primitive graphic symbols, it has been possible to build graphic models, from simple to complex representations of structured knowledge. For example, we can build representations equivalent to conceptual maps, flowcharts (including iterative procedures) and decision trees, and also other types of models useful for educational modeling such as processes, methods and theories. All these types of models have been used in a number of projects since the first publication of the MOT editor in 1999.

Of particular interest are two of these categories of models. The class "processes and methods" includes instructional design methods such as MISA, which we have totally described graphically using the MOT editor, but also the learning scenarios in a course module, represented by multi-actor process graphs. Another 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 use to describe knowledge domains in TELOS application, or to describe the very structure of the TELOS system.

### 1.4. 3. MOT+OWL: A Standardized Ontology Editor

Many ways can be used to describe a domain of study including text-based narratives or informal graphic models,. 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 or knowledge management project to be considered. Still, this very freedom does not 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 that software agents can use to perform different kind of processing. Still, some ambiguity remains. In instructional engineering applications, we had to constrain the MOT graphic language even more to enable the delivery of learning scenarios in a digitized

platform like Explor@-2 [19], Even then, part of the transfer of the design to the delivery platform had to be done manually, to prevent enforcing unnatural graphic representations on the users.

To deliver computer-based learning environments, beyond the phase where informal graphic design has cleared up ideas, we need to move from informal or semi-formal graphs to formal computable graphic representations.

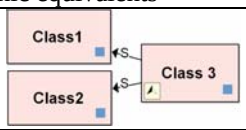
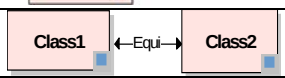
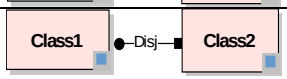
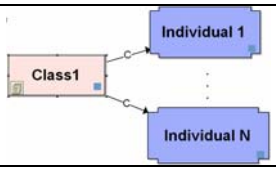
Knowledge in a subject domain can be represented in many ways: taxonomies, thesauri, topic maps, conceptual graphs and ontologies. We have selected to use OWL-DL ontologies [20] for TELOS applications for a number or reasons. It is one of the three ontology Web languages that are part of the growing stack of World Wide Web consortium recommendations related to the Semantic Web. Of these three languages, OWL-DL has a wide expressivity and its foundation in descriptive logic guarantees its computational completeness and decidability. *Descriptive Logic* [21] is an important knowledge representation formalism unifying and giving a logical basis to the well known traditions of frame-based systems, semantic networks, object-Oriented representations, semantic data models, and formal specification systems. It thus provides an interesting framework to represent knowledge.

OWL-DL provides a precise XML schema but no graphic representation per se. Some ontology editors like PROTÉGÉ [22], provide interesting graphical views of an ontology, but the main operations are essentially form-based. Our goal was to provide a complete formal graphic representation of OWL-DL that could combine the virtues of interactive construction and still yield a standard format that can be processed by OWL-DL compliant software.

In the context of the MOT representation system, ontologies, in particular OWL-DL constructs, correspond to a category of models called theories. Ontologies can thus theoretically be modeled graphically using the MOT syntax. While doing this, we found out that although the MOT primitive objects and links were sufficient to represent ontologies expressed in OWL-DL, the graphs would become cumbersome unless new symbols were added. We have thus specialized the MOT language and its graphic editor.

Table 1 gives a few examples of MOT+OWL graphic elements with their interpretation in descriptive logic. In OWL, each of these primitive graphic elements correspond to OWL-DL XML schema components. See [23] for a complete description of the MOT+OWL graphic language.

Table 1 - OWL-DL graphic equivalents

| | |
|---|---|
| **Class intersection**<br>$\forall x$: Class3(x)<br>$\leftrightarrow$ Class1(x) $\wedge$ Class2(x) |  |
| **Equivalent classes**<br>$\forall x$: Class1(x) $\leftrightarrow$ Class2(x) |  |
| Disjoint classes<br>$\forall x$: Class1(x) $\leftrightarrow \neg$Class2(x) |  |
| **Extension of a class**<br>$\forall x$: Class(x)<br>$\leftrightarrow$ (x = Ind 1) $\vee$...$\vee$ (x= Ind N) |  |
| **Functional property**<br>$\forall x,\forall y,\forall z$:<br>Prop(x,y) $\wedge$ Prop(x,z)) $\rightarrow$ y=z |  |
| **Transitive property**<br>$\forall x,\forall y,\forall z$:<br>Prop1(x,y) $\wedge$ Prop1(y,z)<br>$\rightarrow$ Prop1(x,z) |  |
| **Inverse properties**<br>$\forall x,\forall y$: Prop1(x,y) $\leftrightarrow$ Prop2(y,x) |  |

Three types of MOT entities are needed to represent OWL-DL models. Concepts represent classes, principles represent properties and facts represent individuals. On these graphic entities, we add little icons or special links between them. In the standard MOT syntax, these icons or special links would be replaced by principles with "R" links to Classes or Properties. For example, in the second and the two last examples of table 1, the following standard graphs would have the same precise OWL-DL interpretation, but they are less readable and more difficult for human interpretation.

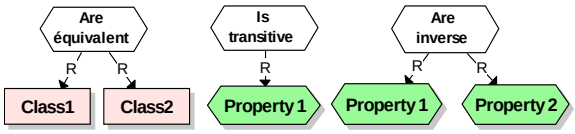



Figure 4 –MOT standard equivalents

Using a limited set of graphic symbols, we can describe formally any semi-formal MOT model that is amenable to a representation in descriptive logic. This is obviously the case for most conceptual models, laws and theory models. Less evident in the case of procedural models, sometimes called task ontologies. Procedural and process/methods models are important for our purpose because learning environments are built around multi-actor processes.

Figure 5 presents a MOT+OWL graph that translates conceptual structure of a learning design presented in the IMS-LD information model [24]. On the figure, "C"

properties are an abbreviation for “is-composed-of” which has the same meaning as the C link in standard MOT models, or the aggregation link in UML models.

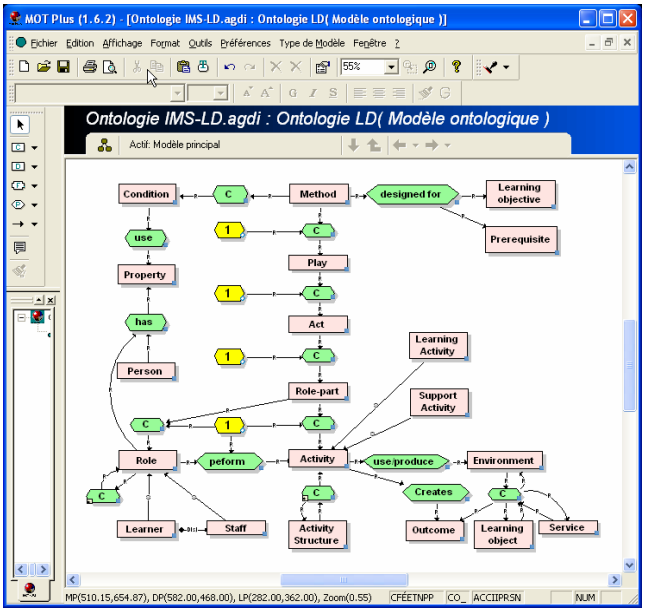



Figure 5 – A simple task ontology for multi-actor scenarios

This example illustrates the fact that functional relations between components of multi-actor processes such as a learning design can be represented by ontologies. Such ontologies have been used to test, for example, the conformance of particular learning designs to the IMD-LD XML schema [25], and to execute them in the context of an ontology-driven system.

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

## 1.5. 4. Representing Multi-Actor Workflows and Learning Designs

Such representations exist in workflows models such as BPMN, the Business Process Modeling Notation [25] and in some instructional design graphic software such as LAMS [26], and our own MISA scenarios using the standard MOT editor. Unfortunately, these representations are either informal like LAMS, semi-formal like MOT, or they are incomplete for learning design modeling, such as BPMM Workflow models. BPMM place all the emphasis on the flow of control in a process, but not on the resources or the knowledge used or produced during the learning delivery process.

To address this problem, we have first developed another MOT specialization: a graphic modeling editor for the IMS-LD specification (level A). Many examples of learning designs have been produced by different groups using this editor[3]. Figure 6 shows part of a simple example of a learning unit on solar astronomy presented recently at a workshop [27]. We see from this example 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 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 C, P, R and I/P links are sufficient to cover all the components of a standard IMS-LD learning design.

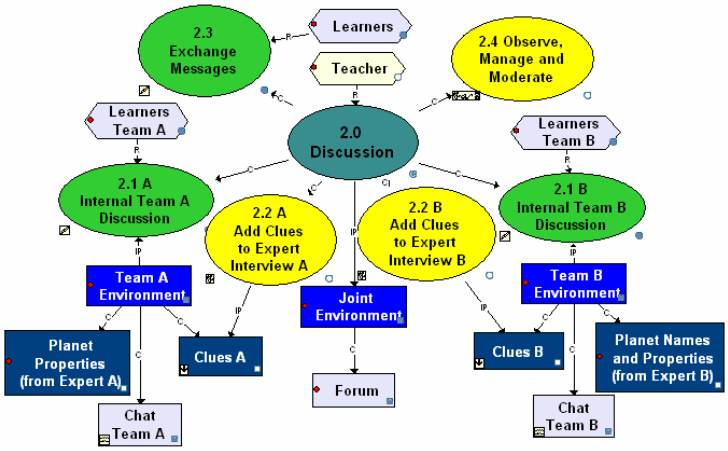



Figure 6 – An example of a MOT+LD learning design

The MOT+LD editor is presented with some detail in [28]. It enables a designer to build graphically a standard IMS-LD model, which is an instance of the IMS-LD task ontology presented on figure 5. Afterwards, the graph is validated and exported as an instance of the IMS-LD XML schema. This XML file can be read in form-based editors such as RELOAD [29], if level B and C conditions or notifications need to be specified. The XML can then be run by IMS-LD compliant players or platforms to deliver on-line learning sessions to their users.

In [30], we have discussed briefly the strengths and weaknesses of the IMS-LD educational modeling specification. One of them is the absence of knowledge representation, which is central to learning and knowledge management that seek to support by the TELOS system. We have proposed to improve that by

[3] These examples are available in the IDLD portal, at www.idld.org.

the semantic annotation of the activities, resources and roles included in a learning design. A *semantic annotation* is a mapping from the ontology to the learning design that associates knowledge elements to the components of the design.

## 5. Towards a function editor for resource aggregation

Another aspect of IMS-LD we need to improve is the control structure of the workflow, that is actually covered by level B and C specifications where properties and conditions can be included in the design to alter the flow of activities, notify an actor or present a resource depending on previous actions or results stored in a user and group file or model. This aspect may not be that important in open learning environments where a total or large degree of liberty is left to the learner and facilitators, but for a business workflow in an organization, or to aggregate software components into larger resources, it is an important dimension.

To address that and provide a basis to build a function editor for the TELOS system, the notion of function maps has been defined as a central piece of the TELOS architecture [31, 32]. Then a comparative analysis has been made between business workflows, IMS-LD learning designs and function maps [33], which has lead to the identification of 21 control situations for workflows encountered in software engineering literature [34].

Based on this work and the MOT+LD editor discussed in section 4, we have designed a new MOT-based graphic editor. The *Function Editor* aims to generalize IMS-LD and capture the main aspects of business workflows. The graphs produced by this editor can be used as executable interfaces (or help define them) for concrete actors to enact the activities and use/produce resources at delivery time. It will also to serve to orchestrate actors, activities and other resources, a fundamental principle built in the TELOS system.

The Function editor uses four kinds of MOT objects with subtypes taken from the TELOS technical ontology [35]. These are shown on figure 7. *Concept* symbols represent all kinds of resources: documents, tools, semantic resources, environments, resource-actors, resource-activities and datatypes. *Procedure* symbols represent activities, including function models or commonly used operation templates to be embedded in other activities. Finally, *principles* are used both to represent different types of actors (as control agents) and control conditions. These two kinds of control entities are represented here by different symbols. The actor's symbols are active agents representing users, groups, roles or software agents that enact the activities using and producing resources as planned by the function model. Conditions are control element inserted within the basic flow to decide on the following activities that can be activated.

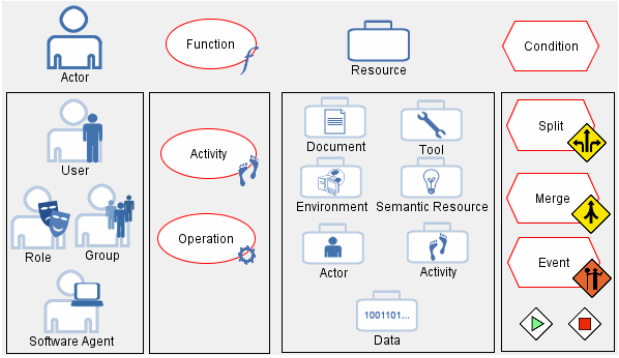



Figure 7– Function Editor Symbols

On figure 8, we see a combination of some of these symbols where a coordinator writes the plan of a document in activity-0. Then three activities are performed by writers. When these are terminated a Web site grouping the different parts is built using a Web editor, and this site is annotated semantically by the group. The basic control flow is shown by P links and it is altered by R links. The data flow is shown by IP links.

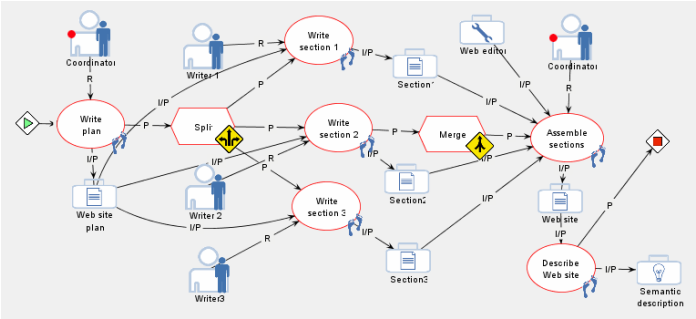



Figure 8– A simple function model

Figure 8 shows a general split condition after activity-0. After that, activities 1, 2 and 3 are executed in parallel, controlled by the properties of the split condition object. Later on, the flow of activities merges through the merge condition object before activity M+1 takes control. This activity will wait for some or all the incoming flows to be activated before it is executed, again based on the properties of the merge condition object.

Figure 9 shows another kind of condition that alters the flow of execution. In activity-2, if a time-event condition is met, the flow of control will not move to activity-3 but to activity-4 and continue, when completed, to activity-5. Properties of the event condition symbol will provide the details on the condition and action parts of the control principle to provide the execution engine with a clear formal definition of the processing to take place.

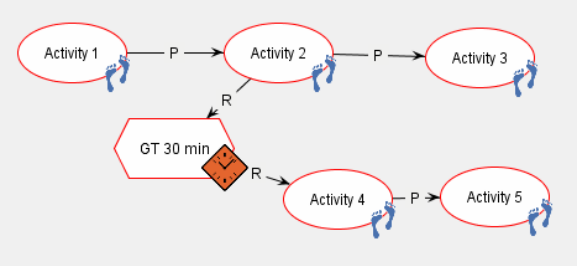



Figure 9– Event-based control

In the Function Editor, we see a combination of a control flow and a data flow. The control flow is modeled using the MOT basic P and R links. P links are used for the basic line of execution. R links identify at which activities an event will trigger a condition to alter the basic flow of control or identify the activities controlled by an actor.

IP links from MOT serve to model the data flow, either from resources to activities where they are consulted, used or processed, or from activities to produced resources. This is why we need to distinguish between actors as active control entities and resource-actors that will serve as data provider or be products of an activity (e.g. a new person of software agent added in a system). A similar distinction is made for resource-activities that can be seen as resources to be transformed, for example by other activities creating or modifying their description.

C links from MOT may also be used to show the composition an entity into other entities A new unification U link is also necessary to guide the execution engine when components are aggregated.

In TELOS, the function editor will enable engineers to combine resources into larger one, technologist to built platform workflows for designers of learning or knowledge management environments, designers to build courses or work designs or scenarios.

Together with an ontology editor that provides semantic annotations of the components of function models, the function editor will enable the graphic definition and execution of the main components of TELOS systems and applications.

## Conclusion: Properties of the Knowledge Representation Paradigm

We now conclude this presentation by discussing the properties of representation languages that have been found most useful.

**Graphic**. The benefits of graphical cognitive modelling have been eloquently summarized by Ausubel, [36], Dansereau [37] and Jonassen [38]. Graphs 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-relation graphs reduce ambiguity compared to a natural language description, but some remain 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.

**User-friendliness.** Not all graphic modeling languages are user-friendly. A good counter-example is UML. The large number models and symbols require considerable expertise for the interpretation and for the construction of the model of a system. Furthermore, each type of model captures a different viewpoint on the information and it is impossible to mix them in the same graph to provide a global view of a subject domain. The representation system must be easy to use without technical or scientific mastery after a short period of initiation. Dansereau and Holley [39] 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 types of links and they are clearly differentiated with well-defined meanings.

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

**Formalizable**. The graphic language should be upward compatible from informal graphs, up to semi-formal and totally unambiguous formal models. At the informal level, an integrated representation framework facilitates thought organization and communication between humans about the knowledge as the graphic representation model evolves. Here the process is more important than the result. At the other end, the graphic language offers more constrained elements to produced totally unambiguous descriptions that can be exported to set of symbols, such as an XML file, that can be processed by computer agents. Here the model is more important than the process.

**Declarative**. Graphic language can be procedural or declarative. Procedural graphic languages have been built in the past, extending flowcharts to promote graphical programming that produces 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. 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 can be used for many different applications, not necessarily the one for which the processing has been planned in a procedural program. This is done by querying the model using an inference engine, in a Prolog-like manner. 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 [40] and it is exactly the way we should see the relation between generic skills and specific domain knowledge in a competency, as meta-knowledge given declaratively, applied to domain knowledge. For example, rules for diagnosing a component-based system applied to models describing a car, a software or a learning environment provide a good way to represent generic skills and competencies.

**Standardized**. Standardization is an important property to enlarge knowledge communication and use between persons and/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. The evolution towards graphic versions of standards like IMS-LD for learning designs and OWL 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 beyond standardization. Not only can the graphic model receive a non-ambiguous formal representation that can be processed by computer agents, but this formal representation is complete (all conclusions are guaranteed to be computable) and decidable (all computations will finish in finite time). These considerations have motivated the construction of the MOT+OWL graphic language that is equivalent to the OWL-DL XML schema based on descriptive logic. OWL-DL ontologies are declarative, and standardized by the W3C.

The team involved in the construction of TELOS[4] is in the process of building a system to assemble learning environments guided by the semantic annotation of resources, including the resources used as system component. TELOS 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. 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.

[4] The TEleLearning Operating System is an assembly workbench to build on-line learning and knowledge management platform, develop in the Canadian LORNET research network, led by the author.

## References


[1] Paquette G. (1996) La modélisation par objets typés: une méthode de représentation pour les systèmes d'apprentissage et d'aide à la tâche. Sciences et techniques éducatives, pp. 9-42, avril 1996

[2] Paquette G. (2002) TeleLearning Systems Engineering – Towards a new ISD model. Journal of Structural Learning 14, pp. 1-35

[3] Chen, P.P.S (1976) The Entity-Relationship model - toward a unified view of data. ACM Transactions on Database Systems I, 1

[4] Sowa, J.F. (1984) Conceptual Structures, Information Processing in Mind and Machine, Addison-Wesley Pubilishing Co, Reading, Mass, 481 pages

[5] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. (1991). Object-Oriented Modelling and Design. Prentice Hall, Englewood Cliffs, New Jersey,

[6] Schreiber G., Wielinga B., Breuker J. (1993) KADS – A Principled Approach to Knowledge-based System Development. San Diego : Academic Press. 457 pp, 1993Sowa, J. (1984)

[7] Booch.G., Jacobson, J. & Rumbaugh, I. (1999) The Unified Modeling Language User Guide. Addison-Wesley, 512 pages.

[8] Merrill M.D (1994). Principles of Instructional Design. Educational Technology Publications, Englewood Cliffs, New Jersey, 465 pages.

[9] Romiszowski A. J. (1981) Designing Instructional Systems. Kogan Page London/Nichols Publising, New York, 415 pages.

[10] Tennyson, R. & Rasch, M. (1988) Linking cognitive learning theory to instructional prescriptions. Instructional Science, 17, pp. 369-385

[11] West West C. K., Farmer J. A., Wolff P. M.. Instructional Design, Implications from Cognitive Science. Allyn and Bacon, Boston, 271 pages, 1991

[12] Inhelder, B & J. Piaget. (1958) The growth of logical thinking from childhood to adolescence. New York: Basic Books

[13] Bruner, J.S. (1973) Beyond the information given. New York, Norton.

[14] Newell A & Simon H. Human problem solving. ( 1972) Englewood Cliffs, NF: Prentice-Hall.

[15] Minski M.. A framework for reprsenting knowledge (1975). In P. H. Winston (ED.), The psychology of computer vision. New York: McGraw-Hill

[16] Paris S., Lipson M.Y., & Wixson K.K. (1983) Becoming a strategic reader. Contemporary Educational Psychology, 8, 293-31Pitrat 1990)

[17] Paquette, G. (1999) Meta-knowledge Representation for Learning Scenarios Engineering. Proceedings of AI-Ed'99 in AI and Education, open learning environments, S. Lajoie et M. Vivet (Eds), IOS Press, 1999.

[18] Paquette (2003) Paquette, G. Instructional Engineering for Network-Based Learning. Pfeiffer/Wiley Publishing Co, 262 pages.

[19] Paquette G. (2001). Designing Virtual Learning Centers. In H. Adelsberger, B. Collis, J. Pawlowski (Eds) Handbook on Information Technologies for Education & Training within the Springer-Verlag series "International Handbook on Information Systems", (pp. 249-272).

[20] W3C (2004) OWL Overview Document (http://www.w3.org/TR/2004/REC-owl-features-20040210/)

[21] Baader, F., D. Calvanese, D.McGuinness, D. Nardi, P,Patel-Schneider, editors (2003) The Description Logic Handbook. Cambridge University Press.

[22] PROTÉGÉ, description and download available at : http://protege.stanford.edu/

[23] Paquette G. and Rogozan D. Primitives de représentation OWL-DL - Correspondance avec le langage graphique MOT+OWL et le langage des prédicats du premier ordre. TELOS documentation. LICEF Research Center. Montreal, Québec.

[24] IMS-LD (2003). IMS Learning Design. Information Model, Best Practice and Implementation Guide, Binding document, Schemas. Retrieved October 3, 2003, from http://www.imsglobal.org/learningdesign/index.cfm

[25] Amorim R., Lama, M. and Sanchez, E (2006) Using Ontologies to model and execute IMS Learning Design Documents, Proceedings othe the 6th IEE International Conference on Adavance Learning Technologies, pp.115-116, Kerkrade, The Netherlands, July 5-7, 2006.

[26] Dalziel, J.R. (2005) LAMS. Learning Activity Management System 2.0. http://wiki.oamsfoundation.org/display/lams/Home.

[26] OMG (2006) Business Process Modeling Notation (BPMM). http://www.bpmn.org/ last retrieve, July 24, 2006

[27] Paquette and Léonard (2006) The Educational Modeling of a Collaborative Game using MOT+LD. Proceedings othe the 6th IEE International Conference on Advanced Learning Technologies, pp.115-116, Kerkrade, The Netherlands, July 5-7, 2006

[28] Paquette, G. M.Léonard, K. Lundgren-Cayrol, S. Mihaila and D. Gareau. (2006) Learning Design based on Graphical Knowledge-Modeling , Journal of Educational technology and Society ET&S , Special issue on Learning Design, January 2006 and Proceedings of the UNFOLD-PROLEARN Joint Workshop, Valkenburh, The Netherlands, September 2005 on Current Research on IMS Learning Design.

[29] RELOAD (2005) RELOAD editor and player, http://www.reload.ac.uk/ last retrieved July 24, 2006

[30] Paquette G. and O. Marino (2005), "Learning Objects, Collaborative Learning Designs and Knowledge Representation", in Technology, Instruction., Cognition and Learning , Vol. 3, p.85-108, Old City Publishing, Inc.

[31] Rosca I (2005) TELOS Conceptual Architecture, version 0.5. LORNET Technical Documents, LICEF research centrer, Télé-université, Montreal.

[32] Paquette, G., Rosca. I, Mihaila S. and Masmoudi A. (2006 in press) TELOS, a Service-Oriented Framework to Support Learning and Knowledge Management, in S. Pierre (Ed) E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective, Springer-Verlag

[33] Marino O. et al., Bridging the Gap between e-learning Modeling and Delivery through the Transformation of Learnflows into Workflows , in S. Pierre (Ed) E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective, Springer-Verlag.

[34] Correal. D., Marino O., Software Requirements Specification Document for General Purpose Function's Editor (V0.4), LORNET Technical Documents, LICEF research centrer, Télé-université, Montreal.

[35] Magnan, F. and Paquette, G. (2006) TELOS: An ontology driven eLearning OS, SOA/AIS-06 Workshop, Dublin, Ireland, June 2006

[36] Ausubel, D. P. (1968) Educational Psychology; A cognitive view. New York, Rhinehart & Winston.

[37] Dansereau D.F. (1978) The development of a learning strategies curriculum. In H. F. O'Neil Jr., (ED.) Learning strategies. New York: Academic Press. Davies,

[38] Jonassen D.H., Beissner K., & Yacci M. (1993) Structural Knowledge – Techniques for Representing, Conveying and Acquiring Structural Knowledge. Laurence Earlbaum Associates, New Jersey, 265 pages.

[39] Dansereau D.F. & Holley, C.D. (1982) Development and evaluation of a text mapping strategy. In A. Flammer & W Kintsch (eds.), Discourse Processing. Amsterdam: North Holland.

[40] Scandura, J.M. (1973) Strutural Learning I: Theory and research. London/New York: Gordon & Breach science Publishers