A Cognitive and Logic Based Model for Building Glass-Box Learning Objects

Philippe Fournier-Viger
University of Quebec at Montreal, Montreal, Canada
fournier_viger.philippe@courrier.uqam.ca

Mehdi Najjar
University of Sherbrooke, Sherbrooke, Canada
mehdi.najjar@usherbrooke.ca

André Mayers
University of Sherbrooke, Sherbrooke, Canada
andre.mayers@usherbrooke.ca

Roger Nkambou
University of Quebec at Montreal, Montreal, Canada
nkambou.roger@uqam.ca

Abstract

In the field of e-learning, a popular solution to make teaching material reusable is to represent it as learning object (LO). However, building better adaptive educational software also takes an explicit model of the learner's cognitive process related to LOs. This paper presents a three layers model that explicitly connect the description of learners' cognitive processes to LOs. The first layer describes the knowledge from a logical and ontological perspective. The second describes cognitive processes. The third builds LOs upon the two first layers. The proposed model has been successfully implemented in an intelligent tutoring system for teaching Boolean reduction that provides highly tailored instruction thanks to the model.

Keywords: learning objects, cognitive modelling, tutoring systems, description logics.

Introduction

Teaching resources are the mean to teach domain knowledge within tutoring systems. In the field of e-learning, a popular solution to increase their reuse is to represent them as learning objects (LOs). The concept of LO is sustained by a set of principles and standards covering many key aspects such as digital rights management, distribution methods and the easiness for adapting existing content. Nevertheless, despite the large efforts to build software to describe and handle LOs, there is no consensus on what a LO is. In fact, some authors proposed very permissive definitions that consider almost any teaching material (digital or non digital) as LOs (LTSC, 1999). As a consequence, tutoring systems generally treat LOs as black boxes. i.e. presented as they are and without individualised feedback for each learner (Simic, Gasevic, & Devedzic, 2004). Moreover, modelling the cognitive processes of learners is fundamental to build educational software that provides highly tailored

instruction (for example, see Anderson, Corbett, Koedinger & Pelletier, 1995). This article presents a model that unify principles of the cognitive modelling theories, which attempts to model the human processes of knowledge acquisition, and principles and standards related to the concept of LO, which takes on the challenges of knowledge engineering, in order to benefit from the advantages of each. LOs described according to our approach are "glass-box learning objects" because they include an explicit description of cognitive processes that enable virtual learning environments (VLEs) to provide highly tailored instruction. The remainder of the article is organised as follows. First, the LO concept is described. Second, the VLE in which the model has been implemented is introduced. Then, the three next sections describe the three layers of our model. We then present preliminary results with the VLE. Finally, the last section announces further work and present conclusion.

The Learning Object Concept

The concept of LO relies on the main idea of structuring learning materials into reusable units. Over the recent years, some definitions of LOs – more or less restrictive – have been proposed in the literature. For example, Duncan (2003) states that the IEEE defines it as "any entity, digital or non-digital, which can be used, reused or referenced during technology supported learning" (LTSC, 1999), Wiley (2002) describes it as "any digital resource that can be reused to support learning" and Koper (2003) adds that "a fundamental idea is that a LO can stand on its own and may be reused". To summarise these definitions, one can state that a LO is an autonomous resource (digital or not) that is reusable in training activities. To clarify their role and their nature, the following subsections describe the five steps of the LOs' lifecycle.

Step 1: Creating an Information Object

The first step of a LO lifecycle consists in creating an information object (IO), i.e. an electronic document of any format (Web pages, images, Java applets, etc.). In general, authors of IOs select the format according to the software with which they wants to ensure compatibility. In e-learning, institutions usually opt for Web documents as IOs to be able to present them via Internet browsers. Among typical examples of IOs: a Web page that explains the process of photosynthesis, an electronic book on linear algebra or a recording of a musical composition by Nicolo Paganini. Furthermore, researchers suggest observing three principles when designing IOs. The latter should be (1) autonomous, (2) adaptable and (3) of low granularity. Creating autonomous IOs means to build objects that are free from reference to external contexts. For example, an author who designs an IO that explains how an engine works must avoid including references to other IOs, because IOs can be presented individually. Nevertheless, authors must use decontextualisation sparingly, because as Wiley, Recker & Gibbons (2000) underline it, decontextualised elements are more difficult to index and have a less clear semantic for a computer. The second principle dictates to create customisable IOs in order to facilitate their integration within particular contexts (Moodie & Kunz, 2003). The third and last principle stipulates that adopting a large granularity reduces the number of IOs that can be assembled together. For example, an e-book is less reusable than its chapters or its paragraphs. Beyond these principles, most institutions propose their own guidelines to create information objects (see, for example, Thomson & Yonekura (2006)).

Step 2: Adding Metadata

The second step of the LOs lifecycle consists in adding metadata to IOs. LOM (LTSC, 2002) is one of the most important metadata standards. To describe an IO, LOM offers about 80 elements grouped in nine categories. Utility of metadata covers three axes. On one hand, metadata facilitate the localisation of IOs stored in repositories. On the other hand, they inform about how to use the

IOs (for example, with regard to copyrights and technology requirements). Finally, they make possible the automatic selection of IOs by a computer (Morales & Aguera, 2002). Metadata are also the element that distinguishes between IOs and LOs. More precisely, appending a learning objective transforms an IO into a LO. This addition ensures that the IO is intended for teaching (Duncan, 2003).

Step 3: Aggregating Learning Objects

The third step is optional in the lifecycle of a LO and consists in joining several LOs in a package to facilitate their distribution and reuse. For example, to simplify their distribution, a professor can group in a package a set of objects that are necessary for a teaching activity. Since a package is also an IO, if an author adds the required metadata, the aggregate will be also considered as a LO. In the popular aggregation standard IMS-CP (IMS, 2005a), a package is a zip file which contains IOs or LOs and a single file which acts as a table of contents.

Step 4: Sequencing Learning Objects

The fourth step is also optional. It consists in determining the presentation order (sequencing) of LOs. Although an author can organise LOs in static sequences, some specifications allow modelling complex learning designs. A learning design defines a teaching activity by means of a set of rules that indicate how to select the next LOs according to the results of intermediate events. Learning designs define also the roles of the various actors who takes part in the activities (learners, support staff, etc.). A widely acknowledged specification for LOs sequencing is IMS-LD (IMS, 2005b), which describes learning designs according to a wide range of approaches of knowledge acquisition such as constructivism and social constructivism. Various learning designs have been done with IMS-LD. For example, the *IMS Learning Design Best Practice and Implementation Guide* (IMS, 2005b) presents a model of an activity where students take part in a simulation of the treaty of Versailles. The actors are learners and the support staff. Their main roles are learner, team leader and teacher. The LOs used are web content resources and the services used are an e-mail server and an online conference software.

Step 5: Delivering Learning Objects

In the e-learning community, the term Learning Management System (LMS) is usually employed to refer to the tutoring systems that present LOs to learners. LMS is a set of software or a Web environment with which training activities that incorporates LOs are carried out (Simic et al., 2004). Many commercial and non-commercial LMS exist, such as WebCT (http://www.webct.com/) and Stellar (http://stellar.mit.edu/). LMS's generally treat LOs as black-boxes. i.e. presented as they are and without individualised feedback for each learners (Simic et al., 2004). The most significant adjustment consists in building a dynamic sequence of LOs (Morales & Aguera, 2002) that the system presents to the learner (for example, a sequence of appropriated Web pages, following the results of a short multiple-choice test). The weak personalisation of LMS's is partially explained by the fact that these tutoring systems often teach full courses and in most cases, they attach great importance to the contribution of human teachers in the learning activities (see Cohen & Nycz (2006) for an overview of the distance education model typically used in LMS's). Building better adaptive educational software also takes an explicit model of the learner's cognitive process related to LOs (Anderson, Corbett, et al., 1995). This article proposes a model for the creation of LOs that include cognitive processes description. Our model organise a domain's knowledge according to three layers, whose each one describes knowledge from a different angle. The first layer defines an ontological and logical representation. The second defines a cognitive representation. The third builds LOs upon the two first layers. The next section introduces the VLE for which the model was tested.

The REDBOOL Boolean Reduction VLE

REDBOOL is a VLE for teaching Boolean reduction. Here, the subject-matter domain is the algebraic Boolean expressions and their simplification by means of reduction rules, which are generally taught to undergraduate students on first cycle of higher education. The tool's purpose is both to help student learn Boolean reduction techniques and to increase confidence with the software. Figure 1 illustrates REDBOOL's main interface. Preliminary notions, definitions and explanations constitute a necessary knowledge background to approach the Boolean reduction problem. This knowledge is available to learners in the "Theory" tab of the VLE. A teaching session consists in solving a sequence of problems. For example, Figure 1 shows the problem to reduce the expression "(((a | F) & (T)) | (~c))". Boolean expressions can be composed of truth constant "T" (true), truth constant "F" (false), proposals "a, b, c, d, e, f" conjunction operator "&", disjunction operator "|" and negation operator "~".The objective of an exercise consists in reducing an expression as much as possible by applying some of the 13 rules of Boolean reduction, such as the disjunction rule of a proposal "a" with the truth constant "False" ((a | F) = (a)), or the De-Morgan rule applied to a conjunction of two proposals (~ (a & b) = (~ a | ~ b)). A learner can select part of the current expression in the "Reduction" field and modify it by means of the keyboard or by using the virtual keyboard proposed. The learner must click on the "Submit step" button to validate changes. In the bottom area of the window, the learner can see the last rules applied. In the top corner on the right side, a progress bar shows the global advancement of the teaching session. The "Advices" section shows the system's feedback (hints, advices, etc.). In the "Examples" tab of the VLE, learners can also ask the system to solve custom problems step by step. The following sections detail each layer of our model with examples from REDBOOL.

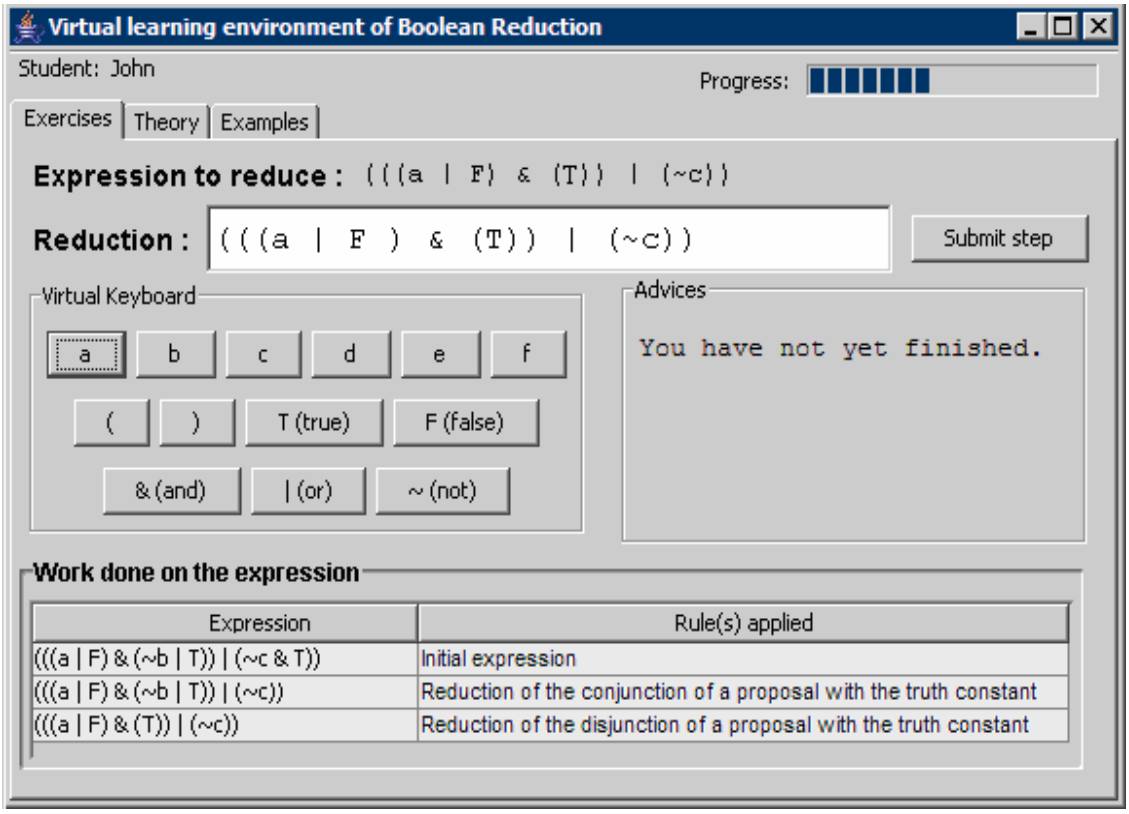


Figure 1: **The REDBOOL VLE**

Layer 1: Logical Representation of the Domain Knowledge

The first layer of our model contains a logical representation of the domain's concepts and their relationships. The formalism used is description logics (DL), a class of knowledge representation languages which exploits - in general - subsets of FOL (see Baader & Nutt, 2003, for a mapping between DL and FOL). We have chosen DL because they are (1) well-studied and (2) they offer reasoning algorithms whose complexity is often lower than those of FOL (Tsarkov & Horrocks, 2003). But the primary reason is that DL employs an ontological approach. i.e., to describe the instances of a domain, they require the definition of (1) general categories of instances and (2) the various types of logical relationships among categories and their instances. The ontological approach is natural for reasoning since even if most interactions happen at the level of instances, most of the reasoning occurs at the level of categories (Russell & Norvig, 2002). As it will be presented, this abstraction fits well with the abstraction between theory and the concrete explanations/exercises/examples found in teaching (Teege, 1994). Additionally, ontologies have shown to be a structure of choice to share, extend and perform reasoning on formal domain descriptions. Building our LOs model upon this theory (1) gives a solid framework to formally interpret the semantic of our LOs individually or jointly and (2) allows benefiting from researches on several topics such as ontology engineering, ontology merging and ontology extraction.

In the DL terminology, whereas a TBOX (terminological box) describes the general knowledge of a field, an ABOX (assertional box) describes a specific world. TBOX contains axioms which relate to *concepts* and *roles***.** ABOX contains a set of assertions which describe *individuals* (instances of concept). Table 1 gives as example a part of the TBOX defined for REDBOOL.

Table 1: Part of layer 1 knowledge for REDBOOL

Concepts	Roles
TruthConstant ⊑ BooleanExpression ⊓ ¬Variable ⊓ ¬DescribedExpression	operator
TruthConstantFalse ⊑ TruthConstant	leftOperand
TruthConstantTrue ⊑ TruthConstant	rightOperand
Variable ≡ BooleanExpression ⊓ ¬TruthConstant ⊓ ¬DescribedExpression	
DescribedExpression ⊑ (BooleanExpression ⊓ (∃operator.⊤ ⊓ ∀operator.Operator)	
Operator ⊑ ¬ BooleanExpression	
NegationOperator ⊑ Operator ⊓ ¬DisjunctionOperator ⊓ ¬ConjunctionOperator	
DisjunctionOperator ⊑ Operator ⊓ ¬NegationOperator ⊓ ¬ConjunctionOperator	
ConjunctionOperator ⊑ Operator	
DisjunctionExpression ≡ DescribedExpression ⊓ ∀operator.OperatorDisjunction ⊓ ∃ leftOperand.⊤ ⊓ ∃ rightOperand.⊤ ⊓∀leftOperand.BooleanExpression ⊓ ∀rightOperand.BooleanExpression	

Atomic concepts and atomic roles are the basic elements of a TBOX. The TBOX of table 1 defines the atomic concepts "TruthConstant", "BooleanExpression", "Variable", "DescribedExpression", "TruthConstant", "TruthConstantTrue", "TruthConstantFalse", "Operator", "NegationOperator", "DisjunctionOperator" and "ConjunctionOperator" (their names begin with a capital letter) and the atomic roles "operator", "leftOperand" and "rightOperand" (their names begin with a small letter). The atomic concepts and atomic roles can be combined by means of constructors to form concept descriptions and role descriptions, respectively. For example, the concept descrip-

tion "BooleanExpression ⊓ Variable" results from the application of constructor ⊓ to atomic concept "BooleanExpression" and "Variable". It is interpreted as the set of individuals who belong to the "BooleanExpression" concept and the "Variable" concept. The various DL are characterised by the set of constructors they propose.

To formally describe constructors' semantic and the TBOX and ABOX notion, it is necessary to define *interpretations*. Because DL follow open-world assumption (the absence of information imply ignorance rather than negative information), the semantic must take into account all possibilities (each interpretation). Formally, an interpretation $\mathcal{I}$ is composed of an interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation domain is a set of individuals. The interpretation function assigns to each atomic concept A a set of individual $A^{\mathcal{I}}$ such that $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each atomic role R; a binary relation $R^{\mathcal{I}}$ such that $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. ABOX assertions are expressed in term of nominals (names that represent individuals). An interpretation $\mathcal{I}$ assigns to each nominal a, an individual $a^{\mathcal{I}}$ from the interpretation domain $\Delta^{\mathcal{I}}$.

Table 2. Concepts constructors, axioms and assertions of the minimal logic $\mathcal{AL}$

Syntax	Semantic	Type
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	Concepts constructors
$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$	
$\exists R.\top$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \exists b^{\mathcal{I}}.(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}\}$	
$\forall R.C$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \forall b^{\mathcal{I}}.(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}} \Rightarrow b^{\mathcal{I}} \in C^{\mathcal{I}}\}$	
$C \equiv D$	$C^{\mathcal{I}} = D$	Axioms
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	
$C(a)$	$a^{\mathcal{I}} \in C$	Assertions
$R(a,b)$	$(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}$	

The first four rows of Table 2 introduce the concepts constructors of a basic DL named $\mathcal{AL}$. The two first columns respectively enumerate constructors' syntax and semantic. The symbols $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ symbolise individual members of $\Delta^{\mathcal{I}}$ for an interpretation $\mathcal{I}$. The letters A and B stand for atomic concepts. The letters C and D represent concepts descriptions. The letters R denote atomic roles. TBOX contain terminological axioms of the form $C \equiv D$ or $C \sqsubseteq D$. The first form states equivalence relations between concepts, whereas the second expresses inclusion relations. The fifth and sixth rows of the table define their semantics. An ABOX contains membership assertions (C(a)) and role assertions (R(a, b)), where a and b are nominals. The two last rows of table 2 explain their semantics. Because assertions are expressed in term of concepts and roles, each ABOX must be associated with a TBOX. An interpretation is said to satisfy a TBOX and an ABOX, if the interpretation imply no contradiction for all assertions and axioms.

The primary purpose of DL is inference. From a DL knowledge base, it is possible to infer new facts, such as deducing all nominals that are members of a concept, finding all concepts D such that $C \sqsubseteq D$, verifying disjointness of two concepts C and D ($C^{\mathcal{I}} \cap D^{\mathcal{I}} = \varnothing$) or checking that a concept C is satisfiable (if an interpretation $\mathcal{I}$ exist, such that $C^{\mathcal{I}}$ is nonempty). Note that several free and commercial inference engines are available such as KAON2 (Motik, Sattler & Studer, 2004), Racer (Haarslev & Möller, 2003) and Pellet (Sirin & Parsia, 2004).

Uses of Description Logics for Domain Knowledge Modeling

To the best of the authors' knowledge, Teege (1994) first proposed to use DL to represent domain knowledge of VLEs. He stated three important originalities. One of them consists of using DL to represent the theory to be taught (TBOX) as an abstraction of natural language (ABOX). This abstraction is necessary to distinguish learners' answers and the VLE's explications/examples in natural language, from the general concepts. In addition, a VLE could extract concepts from natural language answers to form a TBOX; and then compare knowledge of the learners with those of the learning system. Teege demonstrates that inference engines are useful for various tasks such as finding concepts subsuming a misunderstood concept to better explain what characterises it. A second team (Krdzavac, Gasevic, & Devedzic, 2004) cites several ideas from Teege and brings a few novel ones. First, inference could serve to detect modelling errors by inferring the implicit knowledge and by detecting inconsistencies. Secondly, a VLE could interpret a learner's answers as an ABOX and check if it respects the axioms of a TBOX to determine the answers' accuracy. The system could infer why an answer is incorrect. In summary, Teege (1994) and Krdzavac et al. (2004) suggest using DL exclusively for representing the concepts and roles to be taught. They exclude the representation of the *savoir-faire* because DL does not offer means to represent it. However, as Teege commented, "not all of these information needs be represented in DL".

The Role of DL in Layer 1

The first of the three layers that constitute our model is based on DL. It represents concepts of the domain as DL concepts in TBOX, as proposed by Teege (1994). In each defined TBOX, concepts symbolise categories of objects handled in a VLE, and roles represent relationships between these objects. Table 1 shows part of the layer 1 knowledge for REDBOOL. The description has been simplified for the sake of brevity and ease of reading. The first axiom models the concept of Boolean expressions ("BooleanExpression"). The following axioms state that truth constants, variables and described expressions are distinct types of Boolean expressions and specify that there are two subcategories of constants (truth constant "true" and truth constant "false"). Moreover, the example specifies that each described expression has a logical operator that is a conjunction operator, a disjunction operator, or a negation operator. The last concepts axiom asserts that a disjunction expression is a described expression that has a disjunction operator and Boolean expressions as its left and right operands. Roles are defined without further constraints. No ABOX is defined, because it is the level of concrete answers and examples.

The layer 1 knowledge is stored in OWL files, a popular format standardised by W3C to represent knowledge bases for some DL (http://www.w3.org/TR/owl-features). The majority of the inference engines for DL accept OWL files as input. Several authoring tools are also available. We suggest using Protégé (Knublauch, Musen, & Rector, 2004) for its user-friendliness and its support for the almost entire OWL specification. Furthermore, OWL provides several mechanisms to increase the knowledge reuse such as versioning and namespaces. These latter allow to create OWL files that import and extend definitions of existing OWL files without altering the original ones. As a result, authors can split up layer 1 knowledge in several OWL files. As presented further, this facilitates the encoding of knowledge in LOs.

Layer 2: Cognitive Representation of Domain knowledge

Layer 1 structures allow the logical and ontological representation of the domain knowledge. However, building better adaptive educational software also takes an explicit model of the learner's cognitive process. This section presents the layer 2 of our representation model, which fills this gap.

The Psychological Foundations

To structure, organise and represent the knowledge, we have been inspired by cognitive psychology theories, which attempt to model the human process of knowledge acquisition. This knowledge is encoded in various memory subsystems according to the way in which these contents are handled and used. Several authors in psychology, mention – in some form or in another – three types of knowledge. These subsystems are (1) semantic knowledge (Neely, 1989), (2) procedural knowledge (Anderson, 1993) and (3) episodic knowledge (Tulving, 1983). In this paper, we do not discuss the episodic knowledge part of our model since it is the part of our model that records the episodes lived by a person (a history of the use of the two other types of knowledge). However, the interested reader can refer to (Najjar, Fournier-Viger, Mayers & Bouchard, 2005) for detailed explanations.

The semantic memory contains descriptive knowledge. Our model regards semantic knowledge as concepts taken in the broad sense. According to recent researches (Halford, Baker, McCredden & Bain, 2005), humans can consider about four concept occurrences simultaneously (four dimensions) in the achievement of a task. However, the human cognitive architecture has the capacity to group several concepts to handle them as one, in the form of a vector of concepts (Halford, Wilson, Guo, Wiles & Stewart, 1993). We call described concepts these syntactically decomposable concepts, in contrast with primitive concepts that are syntactically indecomposable. For example, in propositional calculus, “a | F” is a decomposable representation of proposal “a”, a non-split representation with the same semantic. The concept “a | F” represents a disjunction between proposal “a” and the truth constant “F” (false), two primitive concepts. The disjunction logical operator “|” is also a primitive concept. In this way, the semantic of a described concept is given by the semantics of its components.

The procedural memory is composed of procedures. i.e. the means to handle semantic knowledge to achieve goals. In opposition to semantic knowledge, which can be expressed explicitly, procedural knowledge is represented by a succession of actions achieved automatically – following internal and/or external stimuli perception – to reach desirable states (Anderson, 1993). Procedures can be seen as a mean of achieving a goal to satisfy a need, without using the attention resources. For example, during the Boolean reduction process, substituting automatically “~T” by “F”, making abstraction to the explicit call of the truth constant negation rule (~T = F, where “T” equals “TRUE”), can be seen as procedural knowledge which was acquired by the repetitive doing. In our approach, we subdivide procedures in two main categories: primitive procedures and complex procedures. Executions of the first are seen as atomic actions. Those of the last can be done by sequence of actions, which satisfy scripts of goals. Each one of those actions results from a primitive procedure execution; and each one of those goals is perceived as an intention of the cognitive system.

We distinguish goals as a special type of semantic knowledge. Goals are intentions that humans have, such as the goal to solve a mathematical equation, to draw a triangle or to add two numbers (Mayers, Lefebvre & Frasson, 2001). Goals are achieved by means of procedural knowledge. In our model, a goal is described using a relation as follows: (R: X, A1, A2 … An). This relation allows specifying a goal “X” according to primitive or described concepts “A1, A2 … An” which characterise the initial state. In a teaching context, stress is often laid on methods that achieve the goal rather than the goal itself; since these methods are in general the object of training. Consequently, the term “goal” is used to refer to an intention to achieve the goal rather than meaning the goal itself. Thus, procedures become methods carrying out this intention (Mayers & Najjar, 2003).

The Computational Representation of the Psychological Model

Layer 2 of our model defines a computational representation of the cognitive model described above. The layer 2 knowledge is stored in files named SPK ("Semantic and Procedural Knowledge"), which describe knowledge entities according to sets of slots.

The concepts' slots

Concepts are encoded according to six slots. The "Identifier" slot is a character string used as a unique reference to the concept. The "Metadata" slot provides general metadata about the concept (for example, authors' names and a textual description). The "Goals" slot contains a goals prototypes list. The latter provides information about goals that students could have and which use the concept. "Constructors" specifies the identifier of procedures that can create an instance of this concept. "Component" is only significant for described concepts. It indicates, for each concept component, its concept type. Finally, "Teaching" points to some didactic resources that generic teaching strategies of a VLE can employ to teach the concept.

The goals' slots

Goals have six slots. "Skill" specifies the necessary skill to accomplish the goal, "Identifier" is a unique name for the goal, "Metadata" describes the goal metadata, "Parameters" indicates the types of the goal parameters, "Procedures" contains a set of procedures that can be used to achieve the goal, and "Didactic-Strategies" suggests strategies to learn how to achieve that goal.

The procedures' slots

Ten slots describe procedures. The "Metadata" and "Identifier" slots are identical to those of concepts and goals. "Goal" indicates the goal for which the procedure was defined. "Parameters" specifies the concepts type of the arguments. For primitive procedures, "Method" points to a Java method that executes an atomic action. For complex procedures, "Script" indicates a list of goals to achieve. "Validity" is a pair of Boolean values. Whereas the first indicates if the procedure is valid and so it always gives the expected result, the second indicates if it always terminate. "Context" fixes constraints on the use of the procedure. "Diagnosis-Solution" contains a list of pairs "[diagnosis, strategy]" that indicate for each diagnosis, the suitable teaching strategy to be adopted. Finally, "Didactic-Resources" points to additional resources (examples, exercises, tests, etc.) to teach the procedure.

Authoring Layer 2 Knowledge

We have developed an authoring tool named DOKGETT (Najjar, Fournier-Viger, Mayers & Hallé, 2005a, 2005b) that permits (1) to model layer 2 knowledge graphically and (2) to generate the corresponding SPK files. The interface (Figure 2) comprises two parts. The left-hand side of the environment consists in a drawing pane where the various knowledge entities can be represented as shapes. Concepts are drawn as triangles. Procedures are represented by circles and goals by squares. Complex procedures and described concepts are delimited by bold contours. Arrows model graphically the relations between knowledge entities. For instance, Figure 2 shows arrows linking a goal to one procedure that can achieve it and to its two concepts argument. The right-hand side of the environment permits the author to specify detailed information about the selected knowledge entity in terms of slots described above.

The Layer 2 Knowledge for REDBOOL

The authoring tool was used to represent the cognitive processes of learners for REDBOOL (Najjar & Mayers, 2004; Najjar, Mayers & Fournier-Viger, 2004) and those of a cook for the re-

alisation of a culinary recipe (Najjar, Fournier-Viger, Mayers, & Bouchard, 2005). As an example, in a single SPK file, we encoded the layer 2 knowledge of REDBOOL. The primitive concepts are truth constant "True", truth constant "False", conjunction operator, disjunction operator and negation operator. The main described concepts are conjunction expression, disjunction expression and negation expression. The file includes procedures and goals for the 13 Boolean reduction rules. It also contains definitions of goals and procedures to create concrete instances of concepts (because each concept's occurrence must be created prior to being handled) and procedures for common errors. In REDBOOL, procedures are fired as a learner operates the graphical interface's buttons (the button/procedure association is found in the "Method" slot of procedures), and the resolution trace is recorded. The VLE connects interactions with the interface to the layer 2 knowledge, and therefore the tutor embedded within the VLE can take decisions on the basis of the cognitive activity of each learner.

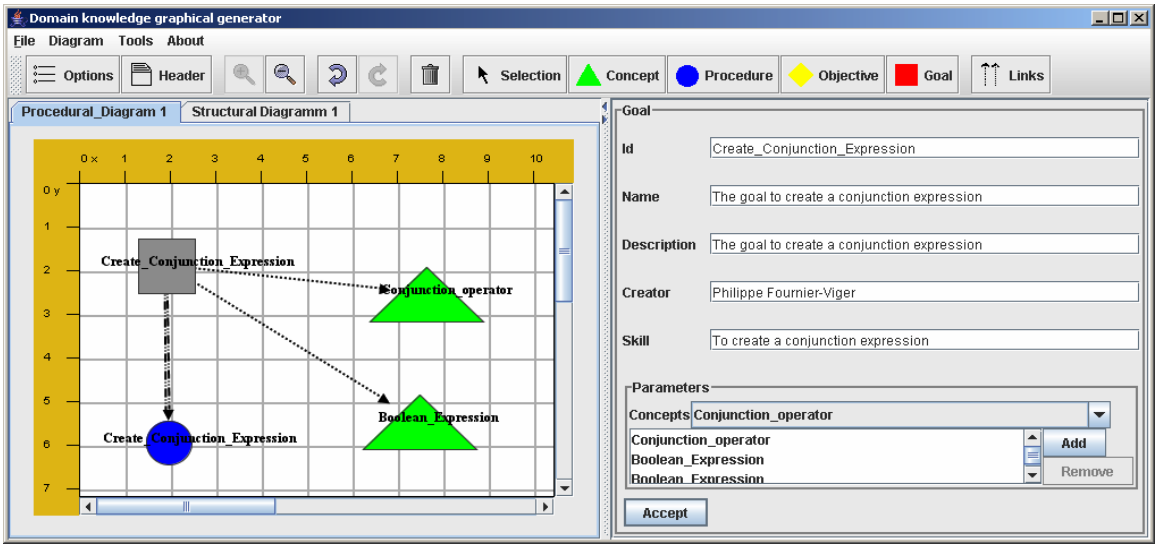


Figure 2: **The DOKGETT authoring tool**

Links between Layer 1 and Layer 2

To establish links between the logical representation of layer 1 and the cognitive representation of layer 2, it is necessary to add additional slots to layer 2 concepts. For this purpose, each primitive concept has a "DLReference" slot that points towards a DL concept. This slot is useful during the instantiation of primitive concepts by procedures. To properly explain the instantiation process of primitive concepts, we will first consider the instantiation of the "F" truth constant. The "Constructors" slot of the "F" truth constant concept states that the procedure "P_CreateTruthConstantFalse" can be used to instantiate the concept. This procedure has its action defined as such. To simulate the instantiation process, our tools adds in an ABOX a nominal associated to the DL concept mentioned in the "DLReference" slot of the concept to instantiate. The table 3 illustrates the resulting assertions added to an ABOX. The nominal "f1" represents a concept instance, and the "TruthConstantFalse(f1)" assertion declare that "f1" is an "F" truth constant. For each instances created, a different nominal is added to the ABOX. In the same vein, the example shows "t1" an instance of the primitive concept "T" truth constant, and "d1", an instance of the disjunction operator primitive concept.

In addition to the "DLReference" slot, each described concept encompasses a slot named "Components", which list one or more roles. Each role associates to a nominal that represent an instance of the described concept, a nominal that represent one of its parts. For example, the nominal "e1" in table 4 correspond to an instance of the described concept "T & F". The "Disjunc-

tionExpression(e1) ” assertion declares that “e1” is a disjunction expression. The “operator(e1, d1)”, “leftOperand(e1,t1)” and “rightOperand(e1, f1)” links the described concept represented by “e1” to nominals that represent its components. Furthermore, a learner can carry out a procedure that replaces a described concept’s component. For instance, when a learner substitute “~T” by “F” in the Boolean expression “a & (~T)”. In this case, the tools we have developed adapt the ABOX accordingly. Therefore, the logical representation of a described concept instance can change and hence its membership to DL concepts.

Table 3: The layer 2 description of the “F” truth constant primitive concept

Slot	Value
Identifier	C_TruthContantFalse
DLReference	TruthConstantFalse
Metadata	Author = Philippe Fournier-Viger
Goals	...
Constructors	P_CreateTruthConstantFalse

Because there is no direct link between layer 2 concepts, correspondence is achieved at the DL level. For this purpose, our tool offer compatibility with three major DL inference engines (KAON2, Racer and Pellet). The absence of link between layer 2 concepts also facilitates the extension of the layer 2 knowledge. Indeed, an author can easily add concepts to any SPK file by associating logical descriptions that extends those of other concepts. Added concepts become automatically compatible with existing procedures and goals. It should be noted that authors can also add new procedures for existing goals, since satisfaction links between a goal and a procedure is stored in procedures’ slots. As a result, authors can create new SPK files that extend existing SPK files without changes.

Table 4: ABOX assertions that represent the “(T & F)” Boolean expression

TBOX	ABOX
As previously defined.	TruthConstantFalse(f1)
	TruthConstantTrue(t1)
	DisjunctionOperator(d1)
	DisjunctionExpression(e1)
	operator(e1, d1)
	leftOperand(e1,t1)
	rightOperand(e1, f1)

Layer 3: Encoding Knowledge as LOs

The third layer builds LOs upon the two first layers. The first step to obtain LOs is creating IOs. According to our model, an IO consists of SPK file(s), OWL file(s), and the VLE. This definition meets the standard definition, which defines IOs as electronic documents. The XML encoding of SPK and OWL files makes files easily customisable. To package files together, we have recourse to IMS-CP, a standard commonly used for LOs (see the first section of this article).

The second step of LOs lifecycle consists in adding metadata to IOs. The IMS-CP specification allows inclusion of metadata in the table of contents of each IMS-CP package, and is compatible with many metadata standards. We use the RELOAD authoring tool (http://www.reload.ac.uk) to specify metadata according to IMS-METADATA, an implementation of the LOM standard. It proposes around 80 attributes to describe an IO. Adding specific metadata attributes is often required for particular e-learning projects (Malaxa & Douglas, 2005). For the needs of our research, SPK files offer additional metadata slots. Two have a great importance. First, the "Include" slot allows importation of definitions from other SPK files by the specification of the SPK file name. The importation mechanism is a key element for LOs –it allows the separation of the knowledge in several files. For example, an author could reuse the procedure of distributing a conjunction over a disjunction "((a & (b | c)) = ((a & b) | (a & c)))" and the procedure of reducing the conjunction of a proposal and its complement "((a & (~ a)) = F)", to specify in a second SPK file the procedure of applying the simplification law "((a & ((~ a) | b)) = (a & b))", a complex procedure that consists in carrying out the two first procedures, one following the other. Second, the "version" slot help ensuring the consistency of the importation mechanism (the slot is included in each SPK file to specify its version). Moreover, according to some authors (Duncan, 2003) transforming an IO into a LO requires the specification of learning objectives that the IO can teach. This addition guarantees that the IO is intended for teaching uses, but more importantly, it indicates the pedagogical use of the IO. The latter information is essential for software or humans that select LOs to be presented. The following paragraphs explain how we express learning objectives in term of the knowledge defined in our SPK files.

Specifying the Learning Objectives

A learning objective is a performance description that the learner must be able to show following training (Gagné, Briggs & Wager, 1992). We consider learning objectives that relate (1) to the acquisition of a skill or (2) to the mastery of a semantic knowledge. First, to check the acquisition of a skill is equivalent to testing the ability to attain a goal. Here, the importance resides in learners' ability to realise the goal. The procedures employed are of no importance, since several correct procedures might achieve the same goal. If a learner accomplishes a goal many times with varied problems and without committing errors, one can conclude that the learner possess the corresponding skill. For example, to check the ability to reduce Boolean expressions, a VLE tutor could provide a set of exercises involving the goal "GoalReduceBooleanExpression". The procedures used to solve the problems are unimportant to the tutor, insofar as they are correct. To test the acquisition of a more specific skill, such as applying the De-Morgan law to an expression of the form (~ (a & b)), the tutor can choose a more specific goal such as "GoalApplyDeMorganLawToNegationOfConjunction". Second, ensuring the mastery of a concept is more complex. Basically, a concept is an inert structure which describes an object. A concept becomes manifest only during a procedure execution which satisfy the goal using that concept. Consequently, a learner must be able to achieve several goals that used the concept in order to show that s/he acquired the concept. For example, to test the acquisition of the concept of truth constant "True", the VLE tutor could test the mastery of the goal to reduce the negation of the truth constant "True" and the goal of simplifying the conjunction of the truth constant "True" and a proposition. This definition of learning objective for a semantic knowledge covers the traditional one of researchers in pedagogy such as Klausmeier (1990), which indicates that mastering a concept require understanding relationships that characterise it. The action of retrieving the relationships can be encoded as procedures. For instance, to master the "canary" concept requires knowing relation "color" between "canary" and "yellow" concept and relation "sub concept" between "canary" and "bird" concept. These relations can be encoded as described concept similar to "(color canary yellow)". An author can add procedures to extract the value from these described concepts and expresses the learning objectives according to the goals associated with these procedures. In

summary, the learning objectives that relate to a skill are expressed in term of a goal to master, whereas those relating to concepts are expressed in term of a set of goal(s) to master. In this sense, our model follows the view of Anderson et al. (1995) that tutoring systems should focus on teaching procedural knowledge.

We propose three slots to represent learning objectives (Fournier-Viger, Najjar & Mayers, 2005). The "Identifier" and "Metadata" slot have the same use as for concepts, goals and procedures. "NecessaryGoals" stipulate goals whose mastery is jointly required to meet the learning objective. Learning objectives are added in the heading of our SPK files. For example, Table 5 presents the learning objective of mastering the concept of truth constant "True". To attain this objective, the "NecessaryGoals" slot states that it is necessary to master the goal of simplifying the negation of the truth constant "True" ((~ T) = F) and the goal of applying the reduction of a conjunction of a truth constant "True" with a proposal "a". i.e., ((a & T) = a). The "EquivalentGoals" slot means that if a learner masters the goal "GoalReduceDisjunctionWithTruthConstant", it is equivalent with regard to the objective as mastering the goal "GoalReduceConjunctionWithTruthConstant". To make these goals equivalent is a pedagogical decision. For instance, an author could have included both as necessary goals.

Table 5: The Learning objective "MasteryOfTheConceptOfTruthConstantT"

Slot	Value
Identifier	MasteryOfTheConceptOfTruthConstantT
Metadata	Author = Philippe Fournier-Viger Creation date = 2006/01/01 Description = The Learning objective of mastering the truth constant "True" concept.
NecessaryGoals	GoalReduceTruthConstantNegation, GoalReduceConjunctionWithTruthConstant,
EquivalentGoals	(GoalReduceConjunctionWithTruthConstant, GoalReduceDisjunctionWithTruthConstant)

Evaluation

A practical experimentation was performed to test the dynamic aspects of our model and especially its ability to represent cognitive activities [13]. We asked ten (10) students in computer sciences and in mathematics who attend the course "MAT-113" or "MAT-114" (dedicated to discrete mathematics) at the University of Sherbrooke to practice Boolean reduction with REDBOOL. An assisted training, aiming to familiarise them with the tool, was given; before leaving them practising. To compare the learners' behaviours, we forced the system to provide them common problems. Parameters of the experiment are reported in Table 6. Exercises complexity ranges from simple (1) to complex (5). For each learner, the system recorded the procedures used as well as the concepts' instances handled. For complexity 1 to 5, the number of goals visited for a complete reduction was about 4, 7, 10, 21 and 40, and the number of concepts' instance manipulated was roughly 4, 14, 24, 35 and 52, respectively. On the whole, we observed that our model well-recorded what characterises the behaviour of each learner. This was noted in the non-predefined combinations between occurrences of concepts and the procedures handling them when achieving goals. Primitive units of semantic and procedural knowledge, chosen with a small level of granularity, are used to build complex knowledge entities, which are dynamically combined – to create a new knowledge – to represent the learner cognitive activity. For example, table 7 presents two different recorded cognitive traces for the exercise "((~b & (~b | F)) & F)". One student answered the problem in two steps, while the other took three steps but made an error at the third step.

Table 6: Parameters of the experiment

Exercise complexity	Number of exercises by student
1 (simple)	4
2	4
3	5
4	6
5 (complex)	6

Following these pilot-tests, we have built a virtual tutor prototype, which takes pedagogical decisions based on simple rules. We have designed it as a proof of concept that the collected data can be use to provide tailored instruction. The prototype tutor reacts when a learner (1) makes a mistake or (2) doesn't reduce a Boolean expression within a specified time limit. In the first case, we consider an error as the result of the learner applying an incorrect procedure for its current goal. In the second case, we consider that the learner either doesn't know any correct procedure for the present goal or doesn't recognize their preconditions. Because our model links goals to procedures that can accomplish them, the tutor has knowledge of all the correct ways to achieve the current goal in both of these situations. Learning and mastering these procedures (or at least one of them) will be one of the immediate objectives of the tutorial strategy. To teach these procedures, the tutor extracts the didactic knowledge encoded in the procedures' slots. For complex procedures that specify sub-goals, the tutor can easily conceive an ordered sequence of valid procedures that allows accomplishing correctly any goal, and gives instruction by making use of the didactic knowledge associated to each procedure. In this virtual tutor prototype, the didactic knowledge consists mostly of short textual hints and explanations. The virtual tutor uses available resources in that order. For example, after analysing the recorded cognitive trace on the right side of table 7, the tutor will detect the use of an erroneous procedure (erroneous procedure #1) and identify that one correct procedure for the same goal is procedure #4 ((a & F) = F). The tutor will first output "Wrong", and then recommend the procedure #4, by giving the associated hint "Look carefully, there is a disjunction expression with the truth constant F." If the learner makes another error or doesn't answer within a time bound, the system advices will become more precise: "You could apply the reduction rule of a disjunction expression with the truth constant F. Here is an example: ((a & F) = F)". This feedback should be sufficient to show to the learner how to correctly solve the problem. But it is not enough to ensure that s/he master procedure #4. Hence, next exercises will be specifically chosen to test procedure #4. Conceiving a more elaborate version of the tutor and verifying its effectiveness is part of our ongoing research.

Table 7: Two cognitive traces to reduce the expression ((~b & (~b | F)) & F)

Student 1 solution		Student 2 solution	
Expression	**Procedure applied**	**Expression**	**Procedure applied**
((~b & (~b | F)) & F)	Initial expression	((~b & (~b | F)) & F)	Initial expression
(~b & F)	Cognitive procedure #14 (a & (a | b)) = a)	((~b & ~b) & F)	Cognitive procedure #14 (a | F) = a)
(F)	Cognitive procedure #4 ((a & F) = F)	(~b & F)	Cognitive procedure #6 ((a & a) = a)
		(~b)	Erroneous procedure #1 ((a & F) = a)

Conclusion and Further Work

Initially, our work will focus on creating LOs following the proposed methodology for new domains. Our team is modelling the knowledge of a virtual laboratory on the fundamental concepts of electric circuits, of a laboratory on the analysis of ADN in genetic engineering, and of another on the A* algorithm in artificial intelligence. Although, our model may need to be improved to represent knowledge of particular ill-defined domains, we expect our cognitive representation to be rather general because similar cognitive models have pretty good results with more than one hundreds domain modeled, ranging from playing Backgammon to simulating driving behaviour (Anderson et al., 2004). Undoubtedly, our current work will lead to improvements of our model. At the logical level, we provide support for the KAON2, Racer and Pellet inferences engines. We are currently investigating the possibility to use SWRL (http://www.daml.org/2003/11/swrl/), an extension of the OWL format with Horn rules, to provide additional expressivity. Recent results in DL field showed that part of SWRL is decidable (Motik et al, 2004). Although, the current performance of inference engines is pretty satisfying, we are working on optimizing their use, to improve performance. In the same vein, we are envisaging the idea of building hybrid reasoning algorithms that can take into account context information to reason more effectively. We are investigating different ways to benefits from the knowledge encoded in our LOs.

In this article, we have proposed an original model for creating reusable units of knowledge that incorporate a logical representation, semantic knowledge, procedural knowledge (the means for manipulating semantic knowledge), as well as didactic knowledge. The model described has been experimented successfully and authoring tools are available for every steps of the modelling process. The inclusion of a logical and ontological structure to describe domain knowledge facilitates the separation of the knowledge in multiple files, their interpretation as a whole, and provides a basis for logical reasoning. Moreover, by using cognitive structures this model permit building LOs that can be used as the basis for providing highly tailored instruction within a VLE. Our model proposes to consider LOs as glass-box LOs rather than black-boxes LOs, as it is often the case in actual LMS's. In this way, our approach constitutes a real improvement over traditional LOs.

References

Anderson, J.R. (1993). *Rules of the mind*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Anderson, J.R, Corbett, A.T., Koedinger, K.R. & Pelletier, R. (1995). Cognitive Tutors: Lessons learned. *The Journal of the Learning Sciences*, *4*(2), 167-207.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. 2004. An integrated theory of the mind. *Psychological Review*, *111*(4), 1036-1060.

Baader, F. & Nutt, W. (2003). Basic description logics. In F. Baader, D. Cavanese, D. McGuinness, D. Nardi, & P. Patel-Schneider (Eds.), *The description logic handbook: Theory, implementation and applications* (pp. 47-100). Cambridge University Press.

Cohen, E. B. & Nycz, M. (2006). Learning objects and e-learning: An informing science perspective. *Interdisciplinary Journal of Knowledge and Learning Objects*, *2*, 23-34. Available at http://ijklo.org/Volume2/v2p023-034Cohen32.pdf

Duncan, C. (2003). Grunarization. In A. Littlejohn (Ed.), *Reusing online resources: A sustainable approach to e-learning* (pp. 12-19). London & New York: Kogan Page.

Fournier-Viger, P., Najjar, M. & Mayers, A. (2005). Combining the learning objects paradigm with cognitive modelling theories - A novel approach for knowledge engineering. *Proceedings of the ITI 3rd International Conference on Information & Communication Technology (ICICT 2005)*. December 5-6, Cairo, Egypt. pp. 565-578.

Gagné, R.M., Briggs, L. Z & Wager, W. (1992). *Principles of instructional design* (4th edition). New York: Holt, Rinehart & Winston.

Haarslev, V., & Möller, R. (2003). Racer: A core inference engine for the semantic web. *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*. October 20, Sanibel Island, Florida, USA. pp. 27-36.

Halford, G.S., Baker, R., McCredden, J.E. & Bain, J.D. (2005). How many variables can humans process? *Psychological Science*, *16*(1), 70-76.

Halford, G.S., Wilson, W.H., Guo, R.G.J., Wiles, J. & Stewart, J.E.M. (1993). Connectionist implications for processing capacity limitations in analogies. In J.K. Holyoak & J. Barnden (Eds.), *Advances in connectionist and neural computation theory: Vol.2 Analogical connections* (pp. 363-415). Norwood.

IMS (2005a). Content packaging specification version 1.2. Retrieved December 14, 2005, from http://www.imsglobal.org/content/packaging/

IMS (2005b) Learning design specification version 1.0 final specification. Retrieved December 14, 2005, from http://www.imsglobal.org/learningdesign/

Klausmeier, H.J. (1990) Conceptualizing. In B.F. Jones & L. Idol (Eds.), *Dimensions of thinking and cognitive instruction* (pp. 20-34), Lawrence Erlbaum Associates.

Knublauch, H., Musen, M.A. & Rector, A.L. (2004). Editing decription logic ontologies with the protégé owl plugin. In V. Haarslev & R. Möller (Eds.), *Proceeding of the International Workshop on Description Logics (DL 2004)*. pp. 70-78.

Koper, R. (2003). Combining reusable learning resources and services with pedagogical purposeful units of learning. In A. Littlejohn (Ed.), *Reusing online resources: A sustainable approach to e-learning* (pp. 46-59). London & New York: Kogan Page.

Krdzavac, N., Gasevic, D. & Devedzic, V. (2004). Description logics reasoning in web-based education environments. *Proceedings of the Adaptive Hypermedia and Collaborative Web-based Systems (AHCWS04)*. Munich, Germany.

LTSC (1999). LTSC Website. Retrieved December 13, 2005, from http://ltsc.ieee.org/wg12/index.html.

LSTC. (2002). IEEE 1484.12.1-2002: Standard for learning object metadata. Retrieved January 1, 2006.

Malaxa, V. & Douglas, I. (2005). A framework for metadata creation tools. *Interdisciplinary Journal of Knowledge and Learning Objects*, *1*, 151-162. Available at http://ijklo.org/Volume1/v1p151-162Malaxa28.pdf

Mayers, A., Lefebvre, B. & Frasson, C. (2001). Miace, a human cognitive architecture. *SIGCUE Outlook*, *27*(2), 61-77.

Mayers, A. & Najjar, M. (2003). Explicit goal treatment in representing knowledge within intelligent tutoring systems. *Proceedings of the 6th International Conference on Computational Intelligence and Natural Computing (CINC 03)*. September 26-30, Cary, North Carolina, USA. pp. 1685-1689.

Moodie, P. & Kunz, P. (2003). Recipe for an intelligent learning management system (ilms). In U. Hoppe, V. Aleven, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo et al. (Eds.), *Supplemental Proceedings of the 11th International Conference On Artificial Intelligence in Education (aied)*. Sydney, Australia. pp. 132-139.

Morales, R. & Aguera, A.S. (2002). Dynamic sequencing of learning objects. *Proceedings of ICALT-2002*. pp. 502-506.

Motik, B., Sattler, U., Studer, R. (2004). Query answering for OWL-DL with rules. *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*. Hiroshima, Japan, November, 2004. pp. 549-563.

Najjar, M. & Mayers, A. (2004). Using human memory structures to model knowledge within algebra virtual laboratory. *Proceedings of the 2nd IEEE International Conference on Information Technology in Research and Education (ITRE 04)*. June 28 – July 1, London, UK. pp. 155-159.

Najjar, M., Mayers, A. & Fournier-Viger, P. (2004). Goal-based modelling of the learner behaviour for scaffolding individualised learning instructions. *Proceedings of the 7th International Conference on Computer and Information Technology (ICCIT-04)*. December 26-28, Dhaka, Bangladesh.

Najjar, M., Fournier-Viger, P., Mayers, A. & Hallé, J. (2005a). DOKGETT – An authoring tool for cognitive model-based generation of the knowledge. *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies (ICALT 05)*. July 5-8, Kaohsiung, Taiwan. pp. 371-375.

Najjar, M., Fournier-Viger, P., Mayers, A. & Hallé, J. (2005b). Tools and structures for modelling domain/user knowledge in virtual learning. *Proceedings of The 16th AACE World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 05)*. June 27 - July 2, Montreal, Quebec, Canada. pp. 4023-4028

Najjar, M., Fournier-Viger, P., Mayers, A. & Bouchard, F. (2005). Memorising remembrances in computational modelling of interrupted activities. *Proceedings of the 7th International Conference on Computational Intelligence and Natural Computing (CINC 05)*. July 21-26, Salt Lake City, Utah, USA. pp. 483-486.

Neely, J.H. (1989). Experimental dissociation and the episodic/semantic memory distinction. *Experimental Psychology: Human Learning and Memory*, *6,* 441-466.

Russell, S.J. & Norvig, P. (2002). *Artificial intelligence: A modern approach* (2nd edition). Prentice Hall.

Simic, G., Gasevic, D. & Devedzic, V. (2004). Semantic web and intelligent learning management systems. *Proceedings of the 2nd International Workshop on Applications of Semantic Web Technologies for E-Learning*. Macéió-Alagoas, Brazil. Retrieved May 15, 2005, from http://www.win.tue.nl/SW-EL/2004/swel-its-program.html

Sirin, E., & Parsia, B. (2004). Pellet: An OWL DL reasoner. *Proceedings of the International Workshop on Description Logics (DL 2004)*. Whistler, Canada.

Teege, G. (1994). Using description logics in intelligent tutoring systems. *Proceedings of the 1994 Description Logic Workshop (DL 2004)*. pp. 75-78.

Thompson, K. & Yonekura, F. (2006). Practical guidelines for learning object granularity from one higher education setting. *Interdisciplinary Journal of Knowledge and Learning Objects*, *1*, 163-179. Available from http://ijklo.org/Volume1/v1p163-179Thompson.pdf

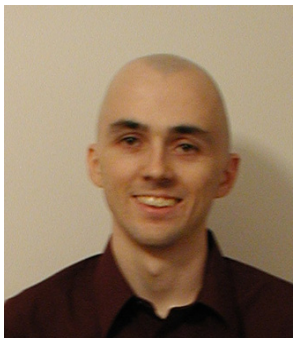
Tsarkov, D. & Horrocks, I. (2003). DL reasoner vs first-order prover. *Proceedings of the 2003 Description Logic Workshop (DL 2003)*. pp. 152-159.

Tulving, E. (1983). *Elements of episodic memory.* New York: Oxford University Press.

Wiley, D.A. (2002). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The instructional use of learning objects (online version).* Retrieved May 15, 2005, from http://reusability.org/read/chapters/wiley.doc

Wiley, D.A., Recker, M. & Gibbons, A. (2000). The reusability paradox. Retrieved May 15, 2005, from http://rclt.usu.edu/whitepapers/paradox.html

Biographies



Philippe Fournier-Viger is beginning a Ph.D. in Cognitive Computer Science at the University of Quebec at Montreal (Canada). He his a member of the GDAC and ASTUS research groups. His research interests are e-learning, intelligent tutoring systems, human-computer interactions, user modeling, knowledge representation, knowledge engineering and cognitive modeling.

Mehdi Najjar received his Ph. D. in Computer Science from the University of Sherbrooke (Canada). He is interested in knowledge representation, management and engineering within virtual learning environments. Being a member of the ASTUS research group, he collaborates with other researchers on the refinement of the knowledge representation structures in intelligent systems.

André Mayers (Ph. D.) is a professor of computer science at the University of Sherbrooke. He founded ASTUS (http://astus.usherbrooke.ca/), a research group about Intelligent Tutoring Systems, mainly focused on knowledge representation structures that simultaneously make easier the acquisition of knowledge by students, the identification of their plans during problem solving activities, and the diagnosis of knowledge acquisition. Andre Mayers is also a member of PROSPECTUS (http://prospectus.usherbrooke.ca/), a research group about data mining.

Dr. Roger Nkambou is currently an Associate Professor in Computer Science at the University of Quebec at Montreal, and Director of the GDAC (Knowledge Management Research) Laboratory (http://gdac.dinfo.uqam.ca). He received a Ph.D. (1996) in Computer Science from the University of Montreal. His research interests include knowledge representation, intelligent tutoring systems, intelligent software agents, ontology engineering, student modeling and affective computing. He is author of more than 80 publications in AIED (Artificial Intelligence in Education) area.