From Black-Box Learning Objects to Glass-Box Learning Objects

Philippe Fournier-Viger[1], Mehdi Najjar[2], André Mayers[2], and Roger Nkambou[1]

[1] Department of Computer Science, University of Quebec at Montreal
{fournier-viger.philippe, nkambou.roger}@uqam.ca
[2] Department of Computer Science, University of Sherbrooke
{mehdi.najjar, andre.mayers}@usherbrooke.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 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.

1 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 that facilitate their reuse and distribution. However, tutoring systems generally treat LOs as black boxes. i.e. presented as they are and without individualised feedback for each learner. Moreover, modelling the cognitive processes of learners is fundamental to build educational software that provides highly tailored instruction [2]. This article presents a model that unify some principles of the cognitive modelling theories, which attempts to model the human processes of knowledge acquisition, and standards related to the concept of LOs, which takes on the challenges of knowledge reuse and distribution. LOs described according to our approach are "glass-box LOs" because they include an explicit description of cognitive processes. The remainder of the article is organised as follows. First, the LO concept is defined. Second, the virtual learning environment (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 a case study where the model has been implemented an evaluated. Finally, the last section announces further work and present conclusion.

M. Ikeda, K. Ashley, and T.-W. Chan (Eds.): ITS 2006, LNCS 4053, pp. 258 – 267, 2006.

2 The Learning Objects Concept

The concept of LO relies on the main idea of structuring learning materials into reusable units. Over the recent years, many definitions of LOs have been proposed. To summarise most of these definitions, one can state that a LO is an autonomous resource (digital or not) that is reusable in training activities [4]. To clarify their role and their nature, this paragraph describes the four steps of the LOs' lifecycle. The first step of a LO lifecycle consists in creating an information object (IO). i.e. an electronic document of any format. E-learning institutions usually opt for Web documents deliverable via Internet browsers. Among typical examples of IOs: a Web page explaining the game of chess or an e-book on linear algebra. Furthermore, authors should avoid creating IOs that include unnecessary references to external contexts, because IOs can be presented individually. IOs should be customizable, to facilitate their integration within particular contexts. Finally, one must determine IOs' granularity carefully, because a large granularity reduces the number of IOs that can be assembled together. For example, an e-book is less reusable than its chapters. The second step of the LOs lifecycle consists in adding metadata to the IOs. LOM (http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html[1]) is one of the most important metadata standards. It offers about 80 attributes to describe an IO. On one hand, metadata facilitate the localisation of IOs stored in repositories. On the other hand, they inform about how to use IOs (for example, with regard to copyrights or technology requirements). Moreover, they make possible the automatic selection of IOs by a computer. Metadata are also the element that distinguishes between IOs and LOs. More precisely, appending a learning objective transforms an IO into a LO, as it ensures that the IO is intended for teaching [4].The third step is optional in the lifecycle of a LO and consists in packaging several LOs to facilitate their distribution (LOs aggregation). For example, a professor can group together a set of objects for a teaching activity. Since a package is also an IO, if one adds the required metadata, the aggregate will be also considered as a LO. In the popular aggregation standard IMS-CP (http://www.imsglobal.org/content/packaging/), a package is a zip file which contains IOs or LOs and a single file which acts as a table of contents. The fourth step of the LOs lifecycle is LOs' delivery. VLEs usually treat LOs as black boxes. i.e. presented as they are without individualised feedback for each learner. The most significant adjustment usually consists in generating dynamic sequences of LOs, following results of questionnaires. The weak personalisation is partly explained by the fact that these tutoring systems often teach full courses and attach great importance to the participation of human teachers. Building better adaptive educational software also takes an explicit model of the learner's cognitive process related to LOs [2]. 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 organise knowledge in LOs. The next section introduces the VLE for which the model was tested.

[1] All URLs mentioned in the paper were last accessed December 14, 2005.

3 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. Fig. 1-a illustrates REDBOOL's interface. Preliminary definitions and explanations are available to learners in the "Theory" tab of the VLE. A teaching session consists in solving a set of problems. For example, Fig. 1-a 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 reduction rules, such as the disjunction rule of a proposal "a" with the truth constant "False" ((a | F) = (a)). A learner can select part of the current expression in the "Reduction" field and modify it 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. The "Advices" section shows the system's feedback (hints, advices, etc.). The following sections detail each layer of our model with examples from REDBOOL.

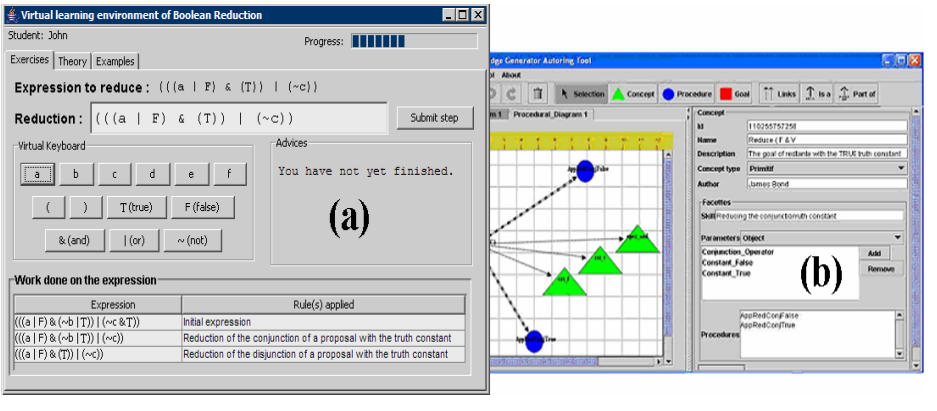

Fig. 1. The REDBOOL VLE (a) and the DOKGETT authoring tool (b)

4 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). We have chosen DL because they are well-studied and widely used logic based languages that offer reasoning algorithms whose complexity is often lower than those of first order logics. DL employ 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 appropriate since reasoning usually occurs at the level of categories. 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	operator
TruthConstantF ⊑ TruthConstant	leftOperand
TruthConstantT ⊑ TruthConstant	rightOperand
Variable ≡ BooleanExpression ⊓ ¬DescribedExpression	
DescribedExpression ≡ BooleanExpression ⊓ (∃operator.⊤ ⊓ ∀operator.Operator) ⊓ ¬ TruthConstant	
DisjunctionExpression ≡ ∀rightOperand.BooleanExpression ⊓ ∃leftOperand.⊤ ⊓ ∀operator.OperatorDisjunction ⊓ ∃rightOperand.⊤ ⊓ ∀leftOperand.BooleanExpression ⊓	

Atomic concepts and atomic roles are the basic elements of a TBOX. Their names begin respectively with an uppercase and a lowercase letter. The atomic concepts and atomic roles can be combined with constructors to form *concept descriptions* and *role descriptions*. For example, the concept description "BooleanExpression ⊓ Variable" results from the application of constructor ⊓ to atomic concept *BooleanExpression* and *Variable*. To formally describe constructors' semantic, it is necessary to define *interpretations*. An interpretation $\mathcal{I}$ consist 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 $A^{\mathcal{I}} \mid A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each atomic role R; a binary relation $R^{\mathcal{I}} \mid R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The concept constructors of a basic DL named $\mathcal{AL}$ [3] are ¬A, ∃R.⊤ and ∀R.C, which are interpreted as $\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$, $\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \exists b^{\mathcal{I}}.(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}\}$ and $\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \forall b^{\mathcal{I}}.(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}} \Rightarrow b^{\mathcal{I}} \in C^{\mathcal{I}}\}$, respectively. The symbols $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ represent individuals that are 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. A TBOX contains terminological axioms of the form C ≡ D or C ⊑ D (defined as $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $C^{\mathcal{I}} = D^{\mathcal{I}}$, respectively). An ABOX contains assertions expressed in term of *nominals*. An interpretation $\mathcal{I}$ assigns to each nominal *a*, an individual $a^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$. An ABOX contains membership assertions (C(a)) and role assertions (R(a, b)), where a and b represent nominals. The assertions' semantics are $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}$, respectively. The primary purpose of DL is inference. From a DL knowledge base, it is possible to infer new facts, such as deducing nominals that are members of a concept, finding all concepts D that subsume a concept C (C ⊑ D), verifying disjointness of two concepts

C and D or checking that a concept C is satisfiable. Note that several free and comercial inference engines are available.

To the best of the authors' knowledge, Teege [11] 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, from the domain 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 or detecting modeling inconsistencies (e.g., unsatisfiable concepts).

The first of the three layers that constitutes our model represents concepts of a domain as DL concepts, as proposed by Teege [11]. In each defined TBOX, concepts symbolise categories of objects handled in a VLE, and roles represent relationships between these objects. Table 1 shows a part of the TBOX for REDBOOL. The first axioms state that truth constants, variables and described expressions are distinct types of Boolean expressions and specify that there are two types of truth constants ("true" and "false"). The last concept axiom asserts that a disjunction expression is a described expression that has a disjunction operator and Boolean expressions as its left and right operands. No ABOX is defined because it is the level of concrete answers and examples. When a learner interacts with our VLE, an ABOX is created dynamically by processes that will be explained afterwards. The layer 1 knowledge is stored into OWL files (http://www.w3.org/TR/owl-features), a popular file format for some DL. Numerous authoring tools and inference engines are available. OWL also offers mechanisms to increase reuse such as versioning and namespaces. Thus, authors can split up layer 1 knowledge in several files. As presented further, this facilitates the encoding of the knowledge in LOs.

5 Layer 2: Cognitive Representation of the Domain knowledge

Layer 1 allows the logical 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 model, which meets this purpose.

5.1 The Psychological Foundations

To structure, organise and represent the layer 2 knowledge, we have been inspired by cognitive psychology theories, which attempt to model the human process of knowledge acquisition. This knowledge is encoded according to the way in which these contents are handled and used. Although there is no consensus on the number of subsystems or on their organisation, the majority of the authors, in psychology, mentions – in some form or in another –semantic knowledge [10], procedural knowledge [1] and episodic knowledge [12]. In this paper, we do not discuss the episodic knowledge part of our model since it is the part that records lived episodes (a history of the use of the two other types of knowledge) for each learner.

The semantic memory contains descriptive knowledge. Our model regards semantic knowledge as concepts taken in the broad sense. According to recent researches [5], humans consider up to 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 [5]. 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 representtation 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., means to handle semantic knowledge to achieve goals. Contrary 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 [1]. 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), 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 [8]. Goals are achieved by means of procedural knowledge. In our model, a goal is described using a relation as follows: (R: X, A_1, A_2 … A_n). This relation allows specifying a goal "X" according to primitive or described concepts "A_1, A_2 … A_n" 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.

5.2 The Computational Representation of the Psychological Model

Layer 2 of our model defines a computational representation of the cognitive model described above. This knowledge is stored in files named SPK, which describe knowledge entities according to sets of 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. 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. Ten slots describe procedures. The “Metadata” and “Identifier” slots are the same as for concepts/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. “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.

We have developed an authoring tool that permits to model and to generate SPK files (Fig. 1-b). The left-hand side of the environment consists in a drawing pane where knowledge entities are represented by different shapes, and arrows represent relations between them. The right-hand side of the environment permits the author to specify detailed information about the selected knowledge entity in terms of the slots described above.

5.3 The Layer 2 Knowledge for REDBOOL

The authoring tool was used to represent the cognitive processes of learners for REDBOOL [9]. As an example, in a single SPK file, we encode 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. The next section explains the link between layer 2 and layer 1, which allow a VLE tutor to make the correspondence between the user activity and the logical description of the domain knowledge found in layer 1.

5.4 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_CreateTruthConstant-False” 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 2 illustrates the resulting assertions added to an ABOX. The nominal “f1” represents a concept instance, and the “TruthConstantF(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.

Table 2. ABOX assertions that represent a “(T & F)” Boolean expression

ABOX
TruthConstantF (f1), TruthConstantT(t1), DisjunctionExpression(e1) , leftOperand(e1,t1), rightOperand(e1, f1), DisjunctionOperator(d1), operator(e1, d1)

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 2 correspond to an instance of the described concept “T & F”. The “DisjunctionExpression(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.

Because there is no direct link between layer 2 concepts, correspondence is achieved at the DL level. 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. 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.

6 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. The XML encoding of SPK and OWL files makes the files easily customisable. To package files together, we have recourse to IMS-CP, a standard commonly used for LOs (cf. section 2.3).

The second step of LOs' lifecycle consists in adding metadata to IOs. IMS-CP packages allow inclusion of metadata compatible with many standards. We use the RELOAD authoring tool (http://www.reload.ac.uk) to specify metadata according to the LOM standard. Moreover, creating a LO requires specifying the learning objectives that it can teach [4]. This addition indicates the pedagogical use of the IO. 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. 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. This definition of learning objective for the semantic knowledge covers the traditional one of researchers in pedagogy. For example, Klausmeier [7], which indicates that mastering a concept require understanding relationships that characterise it. The action of retrieving relationships can be encoded as procedures. In summary, the learning objectives are expressed in term of goal(s) to master. In this sense, our model follows the view of Anderson et al. [2] that tutoring systems should focus on teaching procedural knowledge. We propose three slots for learning objectives. 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 our SPK files.

7 Evaluation

A practical experimentation was performed to test the ability of our model to represent cognitive activities [9]. We asked ten (10) students in computer sciences and in mathematics who attend the course "MAT-113" or "MAT-114" (dedicated to discrete mathematics) 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 this experiment are 1(4), 2(4), 3(5), 4(6), 5(7) where $x(y)$ stands for y exercises of complexity x, for each student. Complexity ranges from simple (1) to complex (5). For each learner, the system noted the procedures used as well as the concepts' instances handled. Analysis of the collected data by a virtual tutor allows it to deduce goals (and subgoals) formulated during the reduction process. 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.

8 Conclusion and Further Work

We have proposed a model for creating reusable glass-box LOs that incorporates logical, cognitive, as well as didactic knowledge (cf. section 5.2). The model has been experimented successfully and authoring tools are available for each steps of the modelling process. The inclusion of a logical structure to describe domain knowledge facilitates the separation of the knowledge in multiple files, and provides a basis for logical reasoning. Moreover, using cognitive structures permit building tutors that presents LOs together with individualised feed-back. In a near future, our work will focus on representing knowledge for new domains. We are also investigating different ways to benefits from the knowledge encoded in our LOs.

References

1. Anderson, J.R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.
2. Anderson, J.R, Corbett, A.T., Koedinger, K.R. & Pelletier, R. (1995). Cognitive Tutors: Lessons learned, *Learning Sciences*, *4*(2), 167-207.
3. Schmidt-Schauß, M. & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1), 1–26.
4. Duncan, C. (2003). Granularization. In Littlejohn, A. (Ed.), *Reusing Online Resources: a sustainable approach to e-learning*. London: Kogan Page.
5. Halford, G.S., Baker, R., McCredden, J.E. & Bain, J.D. (2005). How many variables can humans process?, *Psychological Science*, *16*(1), 70-76.
6. Klausmeier, H.J. (1990) Conceptualizing. In Jones, B.F. & Idol, L. (Eds.), *Dimensions of Thinking and Cognitive Instruction* (pp. 20-34), Erlbaum.
7. Mayers, A., Lefebvre, B. & Frasson, C. (2001). Miace, a Human Cognitive Architecture, *SIGCUE OUTLOOK*, 27 (2), 61-77.
8. Najjar, M. & Mayers, A. (2004). Using Human Memory Structures to Model Knowledge within Algebra Virtual Laboratory. In *Proc. of ITRE 04*. pp. 155-159.
9. Neely, J.H. (1989). Experimental dissociation and the episodic/semantic memory distinction. *Experimental Psychology: Human Learning and Memory*, 6, 441-466.
10. Teege, G. (1994). Using description logics in intelligent tutoring systems. In *Proceedings of the 2004 DL Workshop*. pp. 75-78.
11. Tulving, E. (1983). *Elements of Episodic Memory.* NY: Clarendon Press.