# A Framework for Evaluating Semantic Knowledge in Problem-Solving-Based Intelligent Tutoring Systems

**Philippe Fournier-Viger, Roger Nkambou and André Mayers**

[1]Department of Computer Science, University of Quebec at Montreal
C.P. 8888, Succ. Centre-ville, Montréal, Canada H3C 3P8
[2]Department of Computer Science, University of Sherbrooke
2500 Bld. de l'Université, Sherbrooke, Canada J1K 2R1
{fournier-viger.philippe, nkambou.roger}@uqam.ca

**Abstract**

We describe a framework for building intelligent tutoring systems that offer an advanced evaluation of learners' semantic knowledge. The knowledge model makes a pedagogical distinction between contextual and general semantic knowledge. General knowledge is defined as the knowledge that is valid in every situation of a curriculum, and that a learner should possess. In opposition, contextual knowledge is the knowledge obtained from the interpretation of a situation. Because the model connects the description of general knowledge to the description of procedural knowledge through "semantic knowledge retrieval", the evaluation of general knowledge is not only achieved through direct questions, but also indirectly through observation of problem-solving exercises.

## Introduction

To build e-learning systems that can offer highly-tailored assistance, a well-known approach is to model the cognitive processes of learners according to cognitive theories. The most famous examples of this type of tutoring systems are the Cognitive Tutors by Anderson et al. (1995). They are based on the assumption that the mind can be simulated best by a symbolic production rules system (Anderson 1993). Recently, this idea has been embedded in a development kit named CTAT (Aleven et al 2006). The cognitive Tutors internally describe an exercise with a main goal and a set of applicable rules. Each action done by a learner is seen as the application of a rule that execute an action and can create sub-goals. A rule can be marked correct or erroneous, and be annotated with different tutoring resources. Though these systems obtain great success, they are focused on teaching procedural knowledge (rules) in the context of problem-solving exercises. Anderson et al. (1995) makes this clear: "we have placed the emphasis on the procedural (…) because our view is that the acquisition of the declarative knowledge is relatively problem-free. (…) Declarative knowledge can be acquired by simply being told and our tutors always apply in a context where student receive such declarative instruction external to the tutors. (…) Production rules (…) are skills that are only acquired by doing.".

We claim that this view is limited in two ways. First, it supposes that the declarative knowledge can be taught in an explicit way effectively by human tutors. But, this is not always the case. For some domains the declarative knowledge is best learned by doing. As it will be illustrated here, such domains are the tasks that involve spatial representations. Complex spatial representations are viewed by many researchers as being encoded as semantic knowledge (for example, Tversky 1993). In the remainder of this paper, we adopt the term "semantic knowledge" instead of "declarative knowledge" to designate the declarative knowledge that is not associated with the memory of events (Tulving 1972). Second, Cognitive Tutors cannot evaluate semantic knowledge. They assume that learners will acquire the semantic knowledge before doing the problem-solving exercises, or that it will be available during the exercises and that the learners will know when to use it. The problem is that if a learner possess erroneous semantic knowledge or don't know when to use the semantic knowledge, the Cognitive Tutors will wrongly understand the mistakes made by the learner in terms of procedural errors, possibly triggering inappropriate tutoring behavior.

On the other hand, evaluation of a learner's semantic knowledge in tutoring systems is generally achieved by asking direct questions about that knowledge such as multiple-choice test (for example, Morales & Aguera 2002). Another approach is the automatic scoring of concepts maps that a learner draws by comparing them with an expert map (Taricani & Clariana 2006). A concept map is basically a graph where each node is a concept or concept instance and each link represents a relationship. Our hypothesis is that a more accurate evaluation of semantic and procedural knowledge can be achieved by making explicit the semantic knowledge that a learner should learn, and evaluate it not only with questions but also with procedural knowledge in problem-solving tasks. This is in accordance with educational researchers that emphasize the importance of understanding how the semantic and procedural knowledge are expressed together

in procedural performance (Star 2000). This paper first presents our framework. Then, we describe how it allows the RomanTutor tutoring system to offer tailored assistance.

## A Framework to Support Semantic Knowledge Evaluation

The architecture of our framework (cf. fig. 1(a)) is inspired by those of classical intelligent tutoring systems (Wenger 1987). The main components are (1) the learning activities user interface, (2) the domain knowledge, (3) the student model and (4) the tutoring module. The last two components are domain-independent.

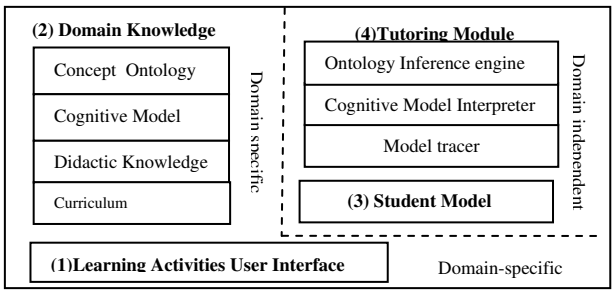


Figure 1: The architecture of our framework

### The Learning Activities User Interfaces

A learning activities user interface has been developed for a tutoring system that assists learners during exercises on the simplification of algebraic Boolean expressions by means of reduction rules (Fournier-Viger et al 2006). However, this paper presents our latest work, applied in the RomanTutor (figure 2). The RomanTutor (Kabanza et al 2005) is a software program for training astronauts to the manipulation of CanadarmII, a robotic arm with seven degrees of freedom, installed on the international space station (ISS). Handling the CanadarmII is a demanding duty since astronauts who control it have a view of the environment rendered by three monitors. Each one show the view usually obtained from a single camera at a time among about ten cameras mounted at different locations on the ISS and on the arm. Guiding the arm requires several skills such as selecting cameras for a situation, visualizing in 3D an environment perceived in 2D and selecting efficient sequences of manipulations. Astronauts follow a rigorous protocol that comprises many steps, because one mistake can engender terrible consequences. The task of interest in RomanTutor is moving the arm from one configuration to another, according to the protocol. To achieve the task, astronauts need to build spatial representations and to visualize them in a dynamic setting. The GUI of RomanTutor reproduces part of the CanadarmII control panel.

### The Concept Ontology

The second step after defining the learning activities interface is to define the domain knowledge. The first part is the concept ontology. The RomanTutor's concept ontology includes about 100 concepts such as the various ISS modules, the main parts of CanadarmII, the cameras on the ISS, and some important strategic 3D zones of the environment surrounding the ISS. The ontology also specifies the important relationships that can exist between instances of concepts. In our framework, the concept ontology inference engine (cf. fig. 1) is the component that offers reasoning services on the concept ontology. The concepts in the concept ontology are linked to the concepts in the cognitive model that is described next. This allows the separation of the cognitive model in multiple files, its interpretation as a whole, and provides a basis for logical reasoning. It permits creating "glass-box learning objects" (see Fournier-Viger et al 2006 for more details).

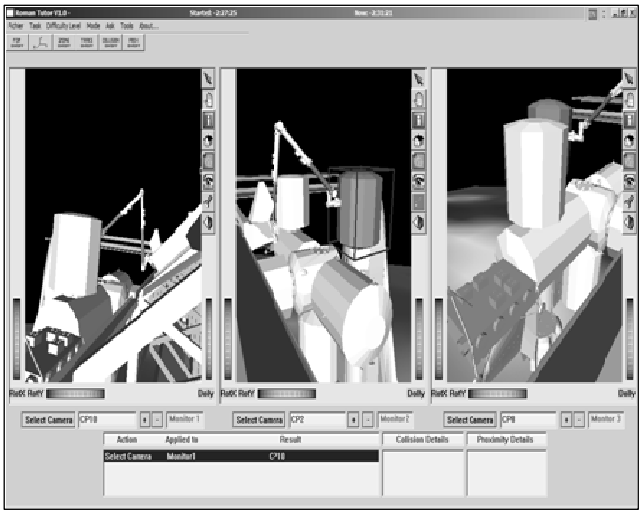

Figure 2: The RomanTutor interface

### The Cognitive Model

After defining the concept ontology, one must describe the cognitive processes relevant to the learning activities. Our cognitive model (Fournier-Viger et al 2006) is a symbolic model that organizes knowledge as semantic knowledge and procedural knowledge. It is inspired by the ACT-R (Anderson 1993) and Miace (Mayers, Lefebvre and Frasson 2001) cognitive theories.

**Semantic Knowledge.** The semantic memory contains descriptive knowledge. Our model regards semantic knowledge as concepts taken in the broad sense. According to recent researches (Halford et al 2005), a human can consider up to four concept instances to perform a task. However, the human cognitive architecture can group several of them to handle them as one (Halford et al 2005). We call "described concepts" these syntactically decomposable representations, in contrast with primitive concepts that are syntactically indecomposable. For example, the expression "PMA03 isConnectedToTheBottomOf Lab02" is a decomposable representation containing three atomic symbols, which represents the knowledge that the "PMA03" ISS module is connected at the bottom of the "Lab02" ISS module on the

ISS (assuming the standard ISSACS coordinate system). In this way, the semantic of a described concept is given by the semantics of its components. While concepts are stored in the semantic memory, concept instances occur in working memory, and are characterized by their mental and temporal context (Mayers et al 2001). Thus, each occurrence of a symbol such as "Lab02" is viewed as a distinct instance of the same concept.

Furthermore, our model distinguishes –on a pedagogical basis -between "general" and "contextual" semantic knowledge. We define general knowledge as the semantic knowledge (memorized or acquired through experience) that is valid in every situation of a curriculum. For instance , such knowledge is that the approximate length of the end effector of CanadarmII is one meter. A general knowledge is a described concept, because to be useful it must represent a relation. To be used properly, general knowledge must (1) be properly acquired, (2) be recalled correctly and (3) be handled by valid procedural knowledge. Conversely, contextual knowledge is the knowledge obtained from the interpretation of a situation. It is composed of concepts instances that are dynamically instantiated during a situation. For example, the information that the rotation value of the joint "WY" of CanadarmII arm is currently 42° is a contextual knowledge obtained by reading the display.

**Procedural Knowledge.** The procedural memory encodes the knowledge of how to attain goals automatically by manipulating semantic knowledge. It is composed of procedures that fires one at a time according to the current state of the cognitive architecture (Anderson 1993). Contrary to semantic knowledge, the activation of a procedure does not require attention. One procedure can be for example, to recall the knowledge of the best camera to view a given ISS module. Another procedure would be to select a camera in the user interface of the RomanTutor. Note that according to the ACT-R theory (Anderson 1993), if two procedures are often applied one after the other to retrieve the same fact, a new specialized procedure can be learned that produce the same result without recalling the fact from long term memory. Furthermore, as Mayers et al. (2001), we differentiate primitive procedures and complex procedures. Whereas primitive procedures are atomic actions, a complex procedure activation instantiates a set of goals, which might be achieved either by a complex procedure or a primitive procedure.

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 et al 2001). At every moment, the cognitive architecture has one goal that represents its intention. There can be many correct and incorrect ways (procedures) to achieve a goal. Our model is based on the proposal of many researchers that goals obey the same constraints as semantic knowledge. i.e. they are competing to become the activated goal, they can be forgotten and their activation vary according to the context. In our model, this assumption means that cognitive steps may not always need to be achieved in the same order. Our model view goals as a special type of described concepts, instantiated with zero or more concept instances that are the object of the goal. For example, the concept instance "Cupola01" could be a component of an instance of the goal "GoalSelectCamerasForViewingModule", which represents the intention of selecting the best camera for viewing the "Cupola01" ISS module. By setting the components of a goal, a procedure determines which procedure(s) can be fired for that goal, and it allows the next fired procedure to access these concepts instances. However, the semantic knowledge passed as goal parameters is often insufficient. A procedure may need to access semantic knowledge stored in the long term memory or working memory. For this reason, our model incorporates a retrieval mechanism similar as the one of the ACT-R theory of cognition to model the recall process. Thus, a procedure can request the retrieval of a described concept by specifying one or more restrictions on the value of its components. A constraint specifies that a component is the same or is different from one of the active goal's components. This mechanism allows, for example, to state that the execution of a procedure request the recall of the described concept "X IsAttachedTo Y" where X is the current goal's camera and Y is the current goal's ISS module. If the retrieval is successful, a concept instance is deposited in a special buffer that accepts the last recalled instance. Then, the next executed procedures can access the concept instance in the buffer to achieve their goal. In our model, a procedure can be specified to fire only if the retrieval buffer contains a specific type of concept instance. Although, our model only allows the retrieval of general knowledge, it allows the definition of primitive procedure that read information from the user interface to simulate the acquisition of contextual knowledge from visual perception.

## The Computational Representation

We have developed a computational structure to describe the cognitive processes for a learning activity according to the cognitive theory described above. The computational structure defines sets of attributes for describing concepts, goals and procedures, respectively. The value of an attribute can be either a set of concept(s), goal(s) or procedure(s), or arbitrary data such as character strings.

Primitive concepts have one main attribute. The "DL Reference" attribute connect the concept to its logical description in the concept ontology. Described concepts have four more attributes. The "Components" attribute indicates the concept type of each concept component. The "General" attribute indicates whether the concept is general or not. For general concepts, "Valid" specifies if the concept is correct or erroneous, and optionally the identifier of an equivalent valid concept (the model allow encoding common erroneous knowledge). Moreover, for general concepts, the "RetrievalComponents" attribute specifies a set of concepts to be instantiated to create the concept components when the concept is recalled through the semantic retrieval mechanism.

Goals have 3 main attributes. “Parameters” indicates the concept types of the goal parameters. “Procedures” contains a set of procedures that can be used to achieve the goal.

Procedures have 7 main attributes. “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. The “Observable” attribute specify for primitive procedures whether the procedure correspond to an action of the user interface or a mental step. For complex procedures, “Script” indicates a set of sub-goals to be achieved with optionally one or more constraints on the order that they should be achieved. “Validity” indicates if the procedure is valid. The attribute “Retrieval-request” can request the recall of a general knowledge. It specifies the identifier of a described concept to be recalled and zero or more restrictions on the value of its components.

## The Cognitive Model Interpreter

We have developed an interpreter to simulate a behavior described with the cognitive model. The interpreter is run by the Model Tracer module (described in the next section). An interpreter state is defined as a set of goals. At the beginning, it contains one goal. At each cycle, the interpreter asks the Model Tracer to choose a current goal among the set of instantiated goals. The interpreter then determines which procedures could be executed for the goal. The Model Tracer must choose a procedure to continue the simulation. Given the execution of a primitive procedure, the interpreter calculates the result, and then removes the goal from the set of goals. Given the execution of a complex procedure, the interpreter instantiate the procedure’s subgoals and add them to the set of instantiated goals. A goal achieved by a complex procedure is only removed from the list of goals, when all its subgoals have been achieved. If the execution of a procedure request the retrieval of a semantic knowledge and one or more choices are possible, the interpreter ask the Model Tracer to choose the knowledge to be retrieved. The interpreter stops when no goals remain.

In the context of learning activities, an author must specify one main goal for each problem-solving exercise. The simulation of solving and exercise with the interpreter give rise to a structure such as the one showed in figure 3.a. On this figure the main goal was achieved by a complex procedure “CP1” which instantiated three subgoals. Whereas, the first goal (“G1”) was realized by the execution of the complex procedure “CP2”, goals “G2” and “G3” were achieved by the observable primitive procedure “OP1” and the complex procedure “CP3”. The two subgoals of the procedure “CP2” were achieved by primitive procedures “PP1” and “PP2”, respectively. The two subgoals of the procedure “CP3” were achieved by the observable primitive procedures “OP2” and “OP3”, respectively. Only the primitive procedures tagged as observable correspond to actions done with the user interface. Figure 3.b show the list of procedures that would be visible for the structure presented on figure 3.a. The next subsection explains how the “Model Tracer” module allows following a student’s reasoning from its visible actions.

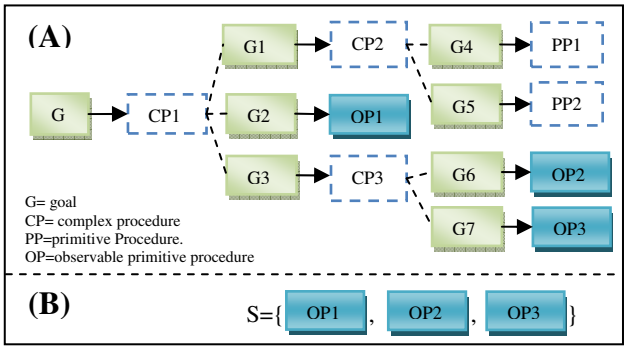


Figure 3: A goals/sub-goals structure for an exercise

## The Model Tracer

The model tracer main task is to find a goals /sub-goals structure such as the one showed on figure 3.a to explain a sequence of learner's action. The input is a list S={p1, p2, … pn} of observable primitive procedures execution along with their arguments. The algorithm proceeds as follows. The interpreter is launched starting from the exercise main goal. When the interpreter offer a choice of several procedures, goals or the retrieval of many different semantic knowledge, the algorithm save the current interpreter state. Then, one possibility is explored and the other possibilities are pushed on a stack to be explored next. In other words, the algorithm explores the state space of the different possibilities in a depth-first way. Every time the algorithm tries a possibility that match a subsequence s⊆S (for example {s1, s2}), but cannot find the next action or if the next action is tagged erroneous, the algorithm notes the current structure together with the subsequence s. After trying all the possibilities, if a structure for S is not found, the goals/sub-goals structure that matches the longest correct subsequence of S is returned. To make the search for the goals/sub-goals structure manageable the algorithm does not explore the possibilities that match a subsequence that is not a subsequence of S, and it does not explore further than the last correct step done by the learner.

## The Student Model

During exercises, the student model update probabilities that indicate the estimated acquisition of each, correct or erroneous, general knowledge or procedure. Procedures are evaluated in problem-solving exercises. The model tracer tries to find a correct goals/subgoals structure for explaining a learner's partial or complete solution. Two types of procedural errors can be detected: (1) the learner applies an erroneous procedure for its current goal or (2)

the learner did not observe the ordering constraints between subgoals for a complex procedure. For example , a learner could forget to adjust a camera zoom/pan/tilt before moving the arm. In this case, the learner forgot to achieve the subgoal of adjusting the camera parameters. Also, we consider that if a learner doesn't react within a time limit, s/he either doesn't know any correct procedure for the present goal or doesn't recognize their preconditions.

### The Tutoring Module

The tutoring module takes pedagogical decisions based on a curriculum. A curriculum is specified as a set of learning objectives along with minimum mastery levels expressed as values between 0 and 1 (Fournier-Viger et al 2006). Different curriculum can be defined to describe different levels of training. A learning objective is a performance description that the learner must be able to show following training (Gagne, Briggs & Wager 1992). A learning objective can be defined as a skill to master. It is a set of one or more goals to master. Such an objective is considered satisfied when the learner shows that s/he masters at least one procedure for each goal according to the student model and the curriculum. The procedures employed are of no importance, since several correct procedures might achieve the same goal. Learning objectives can also be stated in term of one or more general knowledge to master (cf. next section). During a learning session, the tutoring module compares the curriculum requirements with the student model estimates, to generate exercises, questions and demonstrations tailored to the learner that will involve the knowledge to be mastered. Whereas pedagogical knowledge is represented as domain-independent rules in the tutoring module, an author can define "didactic knowledge" (cf. fig. 1), which is domain-specific pedagogical knowledge. In our current work, it consists mostly of pictures or text associated to concepts and procedures. Currently, the pedagogical model is simple. It is part of our current work to make it fully take advantage of the knowledge representation model.

## Evaluating Semantic Knowledge

Our framework provides two ways for evaluating general semantic knowledge. First, it can be tested directly with questions. For instance , RomanTutor may verify the mastery of the described concept "CameraCP9 GivesGlobalViewOf JEM" by showing the learner a view of the "JEM" module and asking him/her to identify which camera was used. Other types of questions are also implemented such as to ask to name the closest modules to a given module, or to ask to select the best cameras for viewing one or more modules. Second, general knowledge can be evaluated through problem-solving exercises. Unlike primitive procedures that are detectable, it is only possible to detect the recall of general knowledge during problem-solving indirectly. Thanks to the goal/sub-goals structure extracted by the Model Tracer, it is possible to know if a general knowledge was recalled. If the learner applies procedures to retrieve a valid knowledge several times, the student model increases its confidence that the learner can recall that knowledge. In the case of the likely recall of an erroneous knowledge, the student model heightens the probability of a recall error with that knowledge and will decrease its confidence that the learner masters the valid concept(s). For example , if the system infers that a learner possesses the erroneous knowledge that camera "CP10" is a good camera to view the JEM module, it will likely generate direct questions about the corresponding valid knowledge or exercises that involve its recall.

To describe the cognitive behavior in the RomanTutor, we have been inspired by results in the field of spatial cognition. In the light of researches carried out during the last decades in neuroscience, experimental psychology and other disciplines, there is no doubt that humans use allocentric and egocentric spatial representations (Nadel and Hardt 2004). An egocentric representation describes the position of an object with regards to oneself. For example, navigating from one place to another with egocentric representations would consist of using a set of stimuli/response associations (Tversky 1993). Usually, this knowledge is gained through experience, but it can also be acquired directly (for instance , from textual route instructions). Route navigation is very inflexible and leaves little room for deviation. Indeed, choosing correct directions with landmarks strongly depends on the relative position of a person to landmarks. Consequently, a path deviation can easily disturb the achievement of the whole navigation task. An incorrect encoding or recall can also compromise seriously the attainment of the goal. Egocentric representations may be sufficient to travel through an environment, but they are inadequate to perform complex spatial reasoning (Tversky 1993). For reasoning that requires inference, humans build cognitive maps that do not preserve measurements but keep the main relationships between elements (allocentric representations). These representations do not encode any perspective but makes it possible to adopt several perspectives. Cognitive maps are also prone to encoding or recall errors. But, it is generally easier to recover from an error, when relying on a cognitive map than on an egocentric representation. Tversky (1993) indicates that a parallel can be drawn between cognitive maps and the semantic memory. Since these latter are key to complex spatial reasoning, tutoring software that diagnose and teach complex spatial reasoning should possess the ability to evaluate semantic knowledge.

According to this view, we have modeled the spatial knowledge for the Canadarm manipulation task as semantic knowledge. To achieve this, we discretized the 3D space into 3D sub spaces named elementary spaces (ES). Spatial relationships are encoded as described concepts such as (1) a camera can see an ES or an ISS module, (2) an ES comprise an ISS module, (3) an ES is

next to another ES, (4) an ISS module is at the side of another ISS module or (5) a camera is attached to an ISS module. The procedural knowledge of how to move the arm from one position to another is modeled as a loop where the learner must recall a set of cameras for viewing the ESs containing the arm, select the cameras, adjust their parameters (zoom, pan, tilt), retrieves a sequence of ESs to go from the current ES to the goal, and then move to the next ES. The model does not go into finer details like how to choose the right joint to move to go from an ES to another. In fact, at this level of details, RomanTutor rely on a special path-planner to track students' errors (Kabanza et al 2005).

## Discussion

One question about our framework is why not just representing semantic knowledge as procedural knowledge? Although it is true that a procedure that request a semantic knowledge from semantic memory and a procedure that use it can be replaced by one or many specialized procedures that perform the same function without recalling the knowledge, this would require to define a procedure for each fact that can be recalled. The benefit of our approach is that the semantic knowledge is explicit. Thus, it can be recalled and taken as parameters by procedures. Hence a task model can have fewer procedures (rules) and these procedures can be more general. In RomanTutor, this showed to be important as it allowed to define the main steps of the CanadarmII manipulation with few procedures, and to annotate these procedures and the general semantic knowledge with specific textual hints. Furthermore, the explicit representation of the general knowledge as relationships is advantageous because it allows evaluating a general knowledge in problem-solving learning activities, but also because it permits generating direct questions about it. For example, to test the semantic knowledge "CP10 canView Zone1", the RomanTutor can generate the questions "? canView Zone1", "CP10 canView ?" and "CP10 ? Zone1". If that general knowledge was not explicit, answering these 3 questions would be viewed as utilizing at least three different specialized procedures and a virtual tutor would not understand that they are related, and that this knowledge can be used in problem-solving exercises.

## Conclusion

We have presented an original framework, which offers a solution for evaluating and teaching general semantic knowledge that learners should possess. Because it connects semantic knowledge to procedural knowledge by semantic knowledge retrieval, evaluation of the general knowledge can be achieved directly through questions or indirectly through observation of problem-solving tasks. Moreover, the virtual tutors based on our framework should be able to generate better feed-back, because they know how the recalled semantic knowledge is connected to procedures. Furthermore, this paper has explained how the framework can be used to evaluate spatial representations by its integration in RomanTutor.

## Acknowledgements

The authors thank the Canadian Space Agency, the Fonds Québécois de la Recherche sur la Nature et les Technologies and the Natural Sciences and Engineering Research Council for their logistic and financial support. The authors also thank Khaled Belghith, Daniel Dubois, Usef Faghihi, Mohamed Gaha and the other current and past members of GDAC/PLANIART teams who have participated in the development of the RomanTutor.

## References

Aleven, V., McLaren, B. M., Sewall, J. and Koedinger, K. 2006. The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. *Proceedings of ITS 2006*, 61-70, Springer Verlag.

Anderson, J.R. (1993). *Rules of the mind*. NJ: Erlbaum.

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

Halford, G.S., Baker, R., McCredden, J.E. & Bain, J.D. 2005, How many variables can humans process?, *Psychological Science*, 16(1):70-276.

Fournier-Viger P., Najjar, M., Mayers, A. & Nkambou, R. (2006). From Black-box Learning Objects to Glass-Box Learning Objects. *Proceedings of ITS 2006*, 258-267.

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

Kabanza F., Nkambou R. and Belghith K. 2005. Path-Planning for Autonomous Training on Robot Manipulators in Space. *Proceedings of IJCAI 2005*.

Mayers A., Lefebvre B & Frasson C. 2001. Miace: A Human Cognitive Architecture. *Sigcue outlook*. 27:61-77.

Morales, R. and Aguera, A.S. 2002. Dynamic sequencing of learning objects. *Proceedings of ICALT 2002*. 502-506.

Nadel, L. and Hardt, O. 2004. The Spatial Brain, *Neuropsychology*, 18(3):473-476.

Star, J.R. (2000). On the Relationship Between Knowing and Doing in Procedural Learning. *Proceedings of the Fourth Intern. Conference of the Learning Sciences,* 80-86.

Taricani, E. M., and Clariana, R. B. 2006. A technique for automatically scoring open-ended concept maps. *Educ. Technology Research & Development*, 54(1), 61-78.

Tulving, E. 1972. Episodic and semantic memory. In Tulving, E. and Donaldson, W. (Eds.), *Organization of memory*. New York: Academic Press, 1972, 381-403.

Tversky, B. 1993. Cognitive Maps, Cognitive Collages, and Spatial Mental Models, *Proc. of COSIT'93*, 14-24.

Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann.